uc3m | Universidad Carlos III de Madrid

Bachelor's Degree in Computer Science and Engineering

2017/2018

Bachelor Thesis

RECOGNITION OF GESTURES THROUGH ARTIFICIAL INTELLIGENCE TECHNIQUES

Rubén Rodríguez Maximiano

Supervisor

Raquel Fuentetaja Pizán

Leganés, July 2th 2018

Resumen

El reconocimiento de gestos consiste en la interpretación de secuencias de acciones humanas captadas por cualquier tipo de sensor, ya sea táctil o no requiera de contacto alguno con el dispositivo, como una cámara. En las últimas décadas ha experimentado un gran avance debido al auge de la Inteligencia Artificial y al desarrollo de sensores cada vez más complejos y precisos.

Un ejemplo concreto ha sido la publicación y el mantenimiento de un SDK oficinal de Microsoft Kinect, con el que los desarrolladores han podido acceder a las capacidades de esta cámara para crear interfaces de usuario más naturales e intuitivas. También se ha incentivado el uso de aplicaciones que van más allá de la industria del entretenimiento, como aquellas que asisten en los cuidados médicos o que permiten la automatización de tareas rutinarias.

Es por ello que en este proyecto hemos desarrollado un conjunto de herramientas para la generación de modelos de aprendizaje capaces de reconocer gestos personalizados para la Kinect v2. El conjunto de herramientas que se ha diseñado e implementado está orientado a facilitar la tarea completa de reconocimiento para cualquier gesto, comenzando con la captura de los ejemplos de entrenamiento, continuando con el pre-procesado y el tratamiento de los datos, y finalizando con la generación de modelos de aprendizaje mediante técnicas de aprendizaje automático.

Finalmente, para evaluar el funcionamiento de la plataforma se ha propuesto y ejecutado una experimentación con un gesto sencillo. Los resultados positivos motivan el empleo de las herramientas desarrolladas para incorporar reconocedores de gestos en cualquier aplicación que utilice el sensor Kinect v2.

Palabras clave Kinect, Reconocimiento de gestos, Aprendizaje Automático, Modelos Ocultos de Markov, Algoritmos de clasificación, Interacción Hombre Máquina

Abstract

The gesture recognition consists of the interpretation of sequences of human actions captured by any type of sensor either touchable or non-touchable like a camera. It has experimented a high progress in the last decades, due to the rise of the Artificial Intelligence and the development of more complex and precise sensors.

One example of this advances was the publish and maintenance of an official SDK of Microsoft Kinect, which were used by developers to access to the capabilities of this camera, so they could create more natural and intuitive applications. This has motivated the use of applications that go beyond the entertainment industry, like those which assists in healthcare or automate routine tasks.

For that reason, this project develops a set of tools for the generation of learning models that are able to recognize personalized gestures for Kinect v2. The set of designed and implemented tools is oriented to ease the task of the recognition of any gesture, starting in the capturing of training examples, continuing with the pre-processing and the treatment of data, and ending with the generation of the recognition models trough machine learning techniques.

Finally, in order to test the functionality of the complete system, an experimentation with a simple gesture has been proposed and executed. The positive results motivate to use the set of developed tools to incorporate gesture recognizers in any application that uses the Kinect v2 sensor.

Key words Kinect, Gesture recognition, Machine Learning, Hidden Markov Models, Classification algorithms, Human-Computer Interaction

Acknowledgments

Gracias a mi familia y amigos por el apoyo recibido durante tantos años, siempre estaré en deuda con vosotros.

Gracias a mis compañeros, que al final amigos, hemos pasado noches de fatiga y estrés pero disfrutado aún más la recompensa del trabajo acabado.

Gracias a mi pareja por irrumpir en mi vida, por soportar los peores momentos e inmortalizar los mejores. Te quiero.

Muchas gracias a mi tutora Raquel por la oportunidad dada, su paciencia y sus buenos consejos.

Acronyms

- **AI:** Artificial Intelligence
- **API:** Application Programming Interface
- **GUI:** Graphical User Interface
- **HCI:** Human-Computer Interaction
- **HMM:** Hidden Markov Models
- **IR:** Infra-red
- **KNN:** K-Nearest Neighbor
- **LOPD:** LEY ORGÁNICA 15/1999, DE PROTECCIÓN DE DATOS DE CARÁCTER PERSONAL
- **ML:** Machine Learning
- **MVP:** Most Valuable Professional
- **PSF:** Python Software Foundation
- **PUI:** Perceptual User Interfaces
- **RBF:** Radial Basis Function
- **SDK:** Software Development Kit
- **SVM:** Support Vector Machines
- **UI:** User Interface
- **UX:** User Experience
- **VR:** Virtual Reality
- **WPF:** Windows Presentation Foundation

Glossary

- **Application:** term used also as 'tool' indistinctly, is a software program designed, developed and used for a certain purpose.
- **Biometric:** related with human body and behavior characteristics.
- **C:** programming language.
- **CSV:** file extension containing comma separated values.
- **C++:** programming language.
- **Data pre-processing:** transformations over a set of data.
- **Dotnet:** software framework, sometimes written as ".NET".
- **Gesture:** human action involving one or more joints of the body.
- **Go:** programming language.
- **Java:** programming language.
- **Joint:** articulation of the human body.
- **Kinect:** camera sensor created by Microsoft. Term used also to refer to the Kinect v2, the second version of the sensor.
- **Python:** programming language.
- **Tool:** term used also as 'application' indistinctly, is a software program designed, developed and used for a certain purpose.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter provides an introduction to the project, describing the motivation points and the main objectives.

Both regulation frame (7) and socio-economic environment (8) have their own chapter later.

1.1 Motivation

Considering the concept definition of 'Gesture recognition' in several technology sites [1] [2] [3] , we can define the term as the mathematical interpretation of human motions by a computer.

The communication between users and computers is called Human-Computer Interaction (HCI) [4], and the interface defines the way this communication is experimented. From the first interfaces based on the use of shell commands, through the combination with keyboard and mouse in Graphical User Interfaces (GUI), interfaces have evolved to ones that focus more on usability and natural interaction. These interfaces have been grouped under the term of Perceptual User Interfaces (PUI) [5], sometimes denominated as Natural User Interfaces (NUI), which purpose is to improve this experience so that the interaction with the system is completely intuitive. The less that the users have to think about how the interface works, the more they are implied in the task they are performing [6]. E.g. if the application verses about editing 3D figures, having the experience of "feeling" that shapes and sizes change with precise hand gestures is more intuitive than using any GUI.

Gesture recognition, as other similar tasks like facial recognition or pose recognition, suppose an ideal component for a natural interaction. Every application that uses NUI has to deal with specific usability domain problems, like sensors and any device that the user has to manage. There will always be learning gaps between users and interfaces, but NUI arise to understand and improve how people communicate with each other and interact with the real world, so it can be applied in the communication with computers.

Any gesture can be defined as a set of rules that describes its path or performance, with a threshold for each step of the trajectory. Though this can be easy for some simple gestures, e.g. hand wave [7], complex gestures involving numerous joints imply defining a complicated set. If

instead of recognizing a gesture, a developer wants to recognize an activity (that implies a sequence of gestures), the problem can become even more difficult. Accuracy is also important when motions from different persons have different values, so thresholds and manual rules may not contemplate every scenario of use. For this reason, using an automatic approach like implementing the models with Machine Learning (ML) can save a lot of time to developers, being possible to adapt fast to different scenarios of use.

1.2 Objectives

This project has the purpose of using Kinect v2 to generate the data for the gesture recognition problem, and then learn models with a Python machine learning library: Scikit-learn. Kinect v2.0 is an upgraded version of the original Kinect sensor, allowing more precision in data capture and adding new hand joints, making even easier to recognize hand gestures. Despite many applications focus on hand and facial gestures, this project considers a general perspective where any gesture can be recognized.

The first step is to record the gestures using Kinect so a gesture dataset is generated.

Secondly, the dataset is processed following the recommendations of *Easy gesture recognition for Kinect paper* [8]. This paper affords the task of creating a gesture recognition tool so developers can include gesture recognizers in their applications. This tool uses two different approaches, Dynamic Time Warping and Hidden Markov Models, and make transformations to normalize and center every joint of every gesture. Otherwise, the same gesture in different positions can be misunderstood if the actors performing the gestures are in different positions or have different heights. These transformations will be analyzed and applied in the current project.

Finally, different classification algorithms will be trained using the gesture dataset: Support Vector Machine (SVM) and K Nearest Neighbors (KNN). The classification will be simplified so the learning models only have to classify between good or bad examples of a concrete gesture. An experimentation planning will be defined so the complete system is tested and both the data pre-processing and the classification algorithms are analyzed.

In order to get a better experimentation and analysis in all the steps of the process, the case study will be simplified as a single joint gesture recognition problem: the dataset will be formed by several instances of a simple gesture, formed by just one body joint. The chosen gesture is a complete clockwise circle performed by right hand.

The complete cycle of generation of the learning models will be approached with the development of the following applications:

- An application to store datasets of gestures generated by Kinect v2, where users can define and perform personalized gestures. The data stored by the application must contain the complete information tracked by the Kinect sensor, so it is possible to have broad experimentation.

- An application to process the datasets according to *Easy gesture recognition for Kinect paper* [8] recommended transformations. The results of applying these transformations have to be analyzed so they are reliable.

- An application to generate learning models using classification algorithms. Different algorithms have to be applied so the experimentation can support the decision of which algorithm fits better for this problem.

Thus, the main objective of this project is to design, develop and evaluate those three applications, and connect them so that the result is a tool useful for performing the full cycle of gesture recognition.

1.3 Document structure

The document of the project is structured into ten chapters, including two annexes, and the list with the consulted references.

The first chapter includes the introduction of the project, describing the motivation, objectives and the current description of the structure of the document.

The second chapter includes the state of the art where the applicable technologies and solution alternatives are contemplated.

The third chapter includes the complete requirement specifications.

The fourth chapter includes a complete description of the development of the project in addition to the experimentation planning and its results.

The fifth chapter includes both conclusions and future lines of the project.

The sixth chapter includes the organization and management followed throughout the development and the planning as too.

The seventh chapter includes the regulation frame applicable to the project regarding the legislation, technical standards, and the intellectual property.

The eighth chapter includes the socio-economic environment where both the socio-economic impact and the project budget are described.

Nine and tenth chapters include appendixes of the applications developed in the project.

Chapter 2

State of the Art

This chapter describes the state of the art of several topics that are fundamental for the project.

Firstly, gesture recognition task is explained with some of the most common applications. The current project is compared with similar existing ones, arguing the value added with the implemented option.

Machine learning is shortly described, giving an insight of the algorithms implied in this project and the machine learning libraries alternatives that were contemplated.

Finally, the Kinect V2.0 is proposed as the sensor to generate the dataset of gestures, but also being compared with other motion sensors.

2.1 Gesture recognition

As it was described in Motivation (see section 1.1), gesture recognition is the interpretation of human motions by a computer. Some definitions of gesture recognition do not include that the interpretation of interactions in touch interfaces is also considered gesture recognition, but it is, as far as users perform gestures with some physical device that the computer has to understand.

A large number of investigations and projects have demonstrated that gesture recognition can benefit HCI and automate tasks that cost a lot of time and effort to professionals. Despite a huge number of applications have emerged from video game industry, more and more applications, not focused on entertainment has appeared. A short summary of them is mentioned next:

- **Security authentication in healthcare systems:** combined with face recognition and biometric systems, makes authentications faster and sterile because any touchable action is required [9].
- **Assistance in medical operations and processes:** not only the sterility is granted with some NUI, but precision in 3D environments so surgeons can operate telematically as they were in the operations room [10]. Surgeons can also manipulate and manage images of the operation performing hand gestures [11].

- **Robot interaction:** combined with face and voice recognition, users can give orders to robots to interact with objects of reality or just start some tasks [10].
- **Biometrics based on behavior:** to identify the person by habits manners like gait or ordinary gestures [12].
- **Analysis of video content:** in order to classify which type of activity is performed in videos. [12].
- **Security and surveillance:** security cameras can track what is happening at any moment and check if current activities need to be watched. This application can be assisted with the recognition of persons based on their behavior [12].
- **Video conference using 3D virtual avatars:** miming face expressions and gestures. Also, higher aspirations on this direction try to create a virtual conference and video chat environment where anyone can see the rest as remotely as they were talking in the same space [13].
- **Rehabilitation assistance for physiotherapists:** storing and analyzing patients progression of their mobility problems. Kinect can support automating several steps of the process so professionals can focus more on results and recovering process [14].
- **Sign language communication between human and computer** [15]
- **Control above driving habits:** adjust vehicle mirrors, audio system, and phone calls can be done with simple hand gestures, avoiding the distraction of the driven task itself [16] [17].

This is just a short list of implemented applications and study cases of gesture recognition. Sites like Kinect Hacks [18] demonstrate that the developer community has been active since Kinect was open to experimentation and progression [13].

In order to have a broader awareness of the gesture recognition applications, a report about trends of gesture recognition has been consulted. This report has studied patents over 5386 records of patents all over the world, and they take from 2003 to 2017 [19].

Figure 2.1 shows the number of patents about gesture recognition applications.

It can be observed that most of the patents verse its application for entertainment purposes, but an important percentage of them is dedicated to automobile and healthcare.

Considering the progress in gesture recognition, gestures cannot be interpreted only as a sequence of raw images but also as a sequence of frames with different representations of data. Microsoft Kinect combines Infra-red (IR) and RGB-D (Depth sensor [20]) to afford the recognition task. It also provides frameworks to get a high-level representation of data: gestures can be described as a sequence of skeleton or body models, where each model have information about joints of the human body.

Kinect device, firstly launched as a gaming peripheral device, has supposed a foster to gesture recognition and similar tasks like posture recognition and facial recognition. It was possible due the sensor was hacked, but also because Microsoft published an official Software Development Kit so developers could access easily to Kinect capabilities.

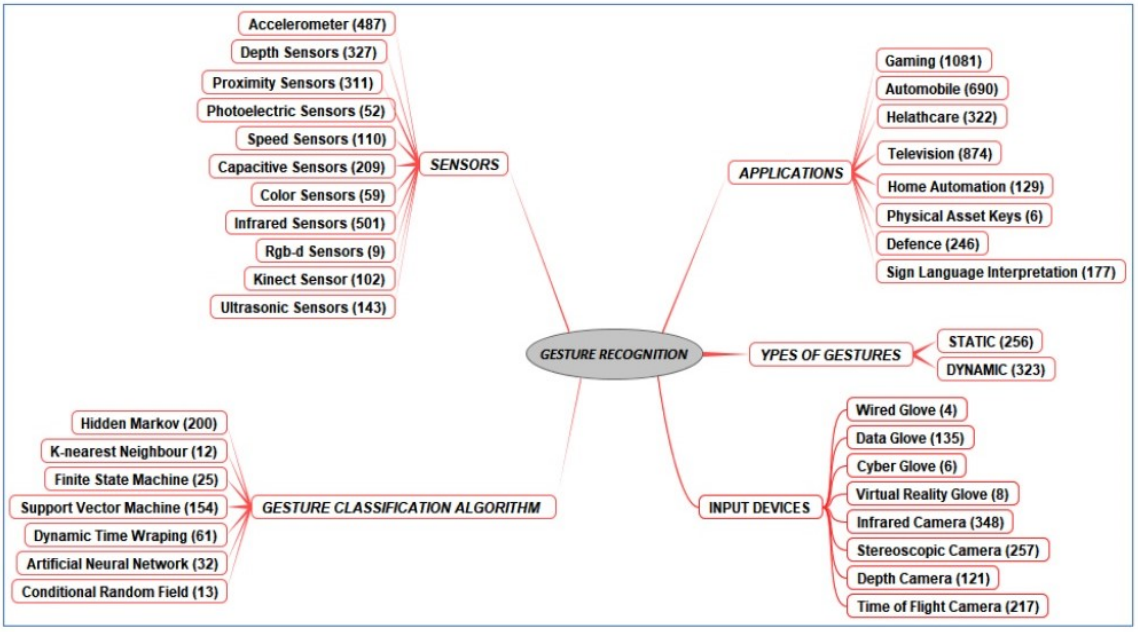

Figure 2.1: Gesture recognition patent categorization

From the Machine Learning perspective, gesture recognition usually implies prediction and classification tasks. Algorithm requirements and data transformations are very dependent on the sensor technology. However, the number of machine learning libraries have grown, implementing a large number of algorithms and facilities. That makes easier to apply machine learning in any application so developers can improve their productivity and focus on the specific characteristics of the problem.

2.1.1 Similar projects

The current project can be compared with others available solutions. Here I present a short list with some of them:

- Vitruvius: created by LightBuzz, is a framework to speed up and facilitates the development of Kinect projects [21]. Its source code is published in Github and its Application Programming Interface (API) offers a natural abstraction of Kinect tracked data and capabilities [22]. Between all the framework capabilities, is worth mentioning the control of 3D avatars though body motions, a wide collection of mathematics transformation over the tracked gestures, access to detailed data of the face and a collection of predefined gestures (mainly involving arms and hands joints). Vitruvius can be used as a gesture recognizer generator, taking advantage of the simple access to Kinect data. It also offers a detailed documentation and blog dedicated to informing about tutorials and advances on the framework. Blog articles are written by Vangos Pterneas, co-founder of Lightbuzz and granted with the title of Most Valuable Professional (MVP) due to his contributions to Kinect open source community. He also has a dedicated blog for Kinect technology and Hololens, helping new developers to make

familiar with Kinect application development [23].

- GesturePak: created by Carl Franklin, Microsoft Regional Director for Connecticut and MVP, consists of an application that can record customized gestures using Kinect v2.0. Those gestures can be edited and then recognized by the application so it brings feedback to the user about the matching of the gesture performance. The gestures are defined with the 25 joints that Kinect v2.0 is able to track [24].

- Gesture Recognition Toolkit: developed by Nick Gillian, the lead of Project Soli in Google's Advanced Technology and Projects group, is an application to train customized gestures. It can be used as a C++ API or as a GUI. This application offers a more generic approach for the gesture recognition problem because the input is independent of any technology sensor [25] [26].

- Visual Gesture Builder: one of the NUI tools that come with Kinect for Windows SDK 2.0, generates recognition models by input XEF files, that describe information of the tracked gesture. With those files, Visual Gesture Builder, using AdaBoostTrigger or RFRProgress, generates a database of models that applications can consult to check if the input action from a user matches with the recognition model [27]. XEF files can be recorded by another tool from the SDK, called Kinect Studio [28].

- EasyGR: is a development tool for the creation of gestures models using machine learning. Its purpose is to help developers to introduce gesture recognition in the application. First developers have to construct the gesture model through a GUI, then export the model to a file, and then call EasyGR library to control the recognition task, so actions can be triggered when the defined gesture is performed. It is compatible with C#. This project has supposed an inspiration for the current one, because of the approach of normalization and center translation of body joints that are tracked through Kinect [8].

Despite the capabilities offered by these alternatives, creating a personalized tool offers the possibility to study deeper the efficiency between ML algorithms and the recognition task. The application is designed to add more learning modules in the future, so it can reused in future investigations.

2.2 Machine Learning

Machine Learning is sometimes confused with the term Artificial Intelligence (AI), but indeed they are different concepts, being AI a broader field of study of computer science, and ML an area of AI. AI objective is to create machines that can act and think like humans, miming the human intelligence. That covers different types of problems like learning, representation of the knowledge or planning. ML focus on the learning task, teaching machines to learn the process of solving tasks by themselves, or just to learn a concrete process. Most of today ML applications have emerged because this field has progressed overwhelmingly thanks to statistics and computer science

combination. Teaching a program on how to understand the process of human communication between humans can be the most scalable solution [29].

Some of the tasks that ML can afford are:

- Classification: given a set of data with label information, a model decides which labels correspond to the new data.
- Regression: given a set of data with continuous values, a model decides which continues values correspond to the new data.
- Clustering: given a set of data without any additional information, a model group this data regarding its characteristics.
- Diagnosis: given a set of observations of one system, a model can discover in which state the system is. The main difference with classification is that data is unlabeled and the output results may be complex.

Machine learning tasks can be categorized into two groups, supervised and unsupervised learning. The input data of the supervised learning is labeled (classification and regression task), while the input data of the unsupervised is unlabeled (clustering and diagnosis).

Gesture recognition may be understood as a classification task, in which the computer has to decide if a certain example represents a specific gesture or not. But it also may be understood as a diagnosis task, in which the input is analyzed so the type of gesture can be deducted. In the case of this project, gesture recognition is proposed as a classification task where the label or class indicates if a gesture is a good or bad example of its execution.

The classification algorithms applied in this project are the Support Vector Machines and the K Nearest Neighbors. The subsection Classification (see subsection 4.5.1) describes the motivation and details of this algorithms for the classification problem of the project. Figure 2.1 shows that both algorithms are included in the list of the most used gesture classification algorithms, being SVM one of the most used.

2.2.1 ML Library

Gesture recognition can be faced with several ML libraries that implement learning algorithms, data transformations, and data representation. Here is presented the list of several libraries and tools that have been taken into account before starting the development of the project:

Python machine learning libraries

- SciPy: library which has algorithms and tools for scientific, technical and mathematical problems. It is a good starting point to know packages that are useful for data representation (NumPy, pandas) and machine learning solutions (Scikit-learn). [30].
- Mlpy: Python module with machine learning algorithms that can be used for supervised and unsupervised problems. It is based on NumPy and SciPy [31].

- Orange: a framework with algorithms for machine learning and data mining. It can be used as a GUI or as a module for Python [32].
- TensorFlow: library with algorithms for mathematics and machine learning, more focused on Deep Learning. It maintains APIs for several languages: Python, C, C++, Java and Go [33].
- Seqlearn: library for sequence classifiers that can operate with Scikit-learn, NumPy, and SciPy. It implements supervised learning with Hidden Markov Models (HMM) and a structured perceptron [34].
- PyStruct: library with machine learning algorithms oriented to structural learning, e.g. SVM, Conditional Random Fields, and Markov Random Fields. Structural learning propose an alternative perspective for problems like classification and regression: they can be modeled as sequence structures or values that have to be predicted [35].
- Scikit-learn: machine learning library for Python that implements algorithms of machine learning, data mining and data analysis. Based on SciPy and NumPy. [36].

.Net/C# machine learning libraries

- numl: library for .NET that includes supervised and unsupervised machine learning algorithms [37].
- AForge.NET: open source framework in C#, used for several areas of AI and Computer Vision [38].
- Accord.NET: framework for .NET oriented to solve audio and video processing problems. Merged with AForge.NET, it offers machine learning algorithms [39].
- Encog: framework with machine learning algorithms for supervised and unsupervised problems (e.g. SVM, HMM, Genetic Programming). It also offers data transformation and processing algorithms and a GUI to make experimentation easier. It can be used with C# and Java [40].

Other frameworks and tools

- Machine Learning Studio (Azure): service to experiment with machine learning solutions in the cloud, being also possible to transform data [41].
- Weka: a collection of algorithms for machine learning and data mining. It can be used as an external tool or called by Java code [42].

Data representation and transformation

- NumPy: library to represent data as N-dimensional arrays, with functions for transformation and analysis. Several mentioned libraries accept this structure for the input data of the algorithms: Scikit-learn, Seqlearn, Mlpy [43].

- Pandas: similar to Numpy, used to make richer representation of data, more flexible and more abstract, and also to make analysis and transformations over it. It can be used in Scikit-learn for the input data [44].

The main reason to consider .NET and C# libraries is that the application that extracts and records the gestures from Kinect is written in Windows Presentation Foundation (WPF): a system for Windows-based applications based on graphical user interfaces that forms part of the .NET framework [45]. The application in charge of data transformation and machine learning task can be integrated with the WPF application easier if they all bellow the same ecosystem.

By another hand, Python offers a broad range of possibilities of libraries for the project purpose. Considering the availability of tutorials and documentation, as the good the good synchronization with Numpy and Scipy, I have selected Scikit-learn as the library for the machine learning application [46]. But also PyStruct and Seqlearn are proposed as alternatives that contemplate structured prediction and have similarities with Scikit-learn API. As far as the machine learning tool encapsulates the interaction with Scikit-learn API, adding new algorithms from different libraries is not an impediment.

2.3 Kinect sensor

Video game industry has always been experimenting with new interfaces, and especially those which try to make a more immersed experience. Following the success of engineering investigations, the game industry has introduced kinetics gloves, camera sensors and even motion controllers to define the game interaction. The popularization of Wii motion has introduced new groups of users that usually never played games. This shows that the use and improvement of NUIs can decrease the learning gap to start playing. Motion recognition has to focus more on velocity but maintaining good levels of accuracy, so users perceive a real-time system reaction and maintain the game experience. [10].

Wii Remote, the controller of Nintendo Wii console, was hacked in 2007 by reverse engineering, opening the path for developers to build their own applications and games. This community has grown considerably so sites like WiiBrew has emerged as a wiki of homebrew development of the Wii console [47] [48]. A project to recognize gestures with Wii Remote tackle the problem using wiigee (developed and published by the authors), designed to capture and data specifically performed by the Wii controller, so customized gesture can be modeled [49] [50].

As Nintendo and Microsoft offered their own natural interaction device (Wii motion and Kinect), Sony launched PlayStation Move for PlayStation 3, allowing users to play through motions with a physical device, similar to Wii remote. PlayStation announced Move.me, a server application that allows developers to access the data captured by PlayStation Move controller [51]. This can be bought by PlayStation Store nowadays [52] , but dedicated official site (`www.us.playstation.com/move-me`) is not accessible. After that, PlayStation 4 continued the support and the development of PlayStation Move, then Sony acquired Softkinetic company, which offered solutions for gesture recognition problems with time-of-flight cameras technology [53]. Currently, Sony offers several libraries for gesture recognition using that technology. One of then,

Depthsense HTlib, using PlayStation Camera, is designed to track motion and body information of users (maximum tracking of 4 users at the same time). But it is not public, being a request needed to start using the library [17].

Microsoft launches Kinect sensor in 2010, a beta SDK in 2011, and a stable API in 2012 (also known as Kinect for Windows). This SDK tracks information of 21 body joints, accessing to position and rotation data and allows face recognition. In 2013 a new version of the camera is announced, allowing 25 body joints (adding more joints of hands), improving reliability and allowing the maximum of 6 users being tracked at the same time. This new version was followed by a new version of the SDK [54] (see Figure 2.2).

Figure 2.2: Kinect v2 sensor.

Leap Motion, Inc. has developed a hand tracking sensor with high accuracy and sensibility. Based on IR and LEDs technology, does not require any contact with users and it is being used for Virtual Reality (VR) applications. It offers SDKs for several languages and technologies, e.g. Unity, C++, Python, and a Beta upgraded version of the previous API [55] [56] [57].

The uSens company offers Fingo, a peripheral for hand tracking that can be used in VR games and applications. It offers a well-documented SDK for Unity, covering the recognition of around 25 hand gestures, but gives also access to bones rotations and relative positions (in relation with close bones) [58].

Nuitrack has several sensors for body tracking: VicoVR and Orbbec Astra, both using the Nuitrack SDK. The SDK allows developers to afford gesture recognition tasks, communication with Unity and Unreal, and also link with many devices: Kinect v1, Asus Xtion, Orbbec Astra, Orbbec Persee and Intel RealSense [59] [60].

Considering the range of devices to choose in order to develop a tool for gesture recognition, Kinect v2 is the selected device due the price, the well-documented SDK and precise sensors. Other reasons to discard some of the other devices are:

- Wii motion can be discarded because it does not offer an official library or SDK.
- PlayStation Move does not offer a free SDK.
- Leap Motion and uSens only track hands information.

Nuitrack seems to be the best alternative to Kinect, due the similar technical specifications shared between the compatible devices. But the scope of this project is to create a tool to generate recognizer models, so using Kinect SDK reduce the problem to one single device.

Even though there are many SDKs for Kinect, this project uses the Microsoft supported Kinect for Windows SDK 2.0 [54].

Chapter 3

Requirements and user stories

The project has followed an iterative and incremental development, combined with Personal Kanban Agile framework, described in Project organization and management (see chapter 6). To synchronize with this methodology, requirements has been replaced by user stories. A user story is a requirement representation that permits an easy conversation of the requirements described inside.

User stories can be defined in different levels of detail, being common the following taxonomy:

- Epic: big user story with high level of abstraction, usually defined at the beginning of the development. In this project, epics focus on the three main applications required to complete the gesture recognition process.
- Theme: user story that arises from the details of epics. Here the themes describe the functionality of each application.
- User story: most detailed type of user story, that appears when themes are detailed. Non-functional requirements are included in this category since they refer to low level details in the abstraction hierarchy. To avoid confusion, this user stories will also be called "little user stories".

One advantage of using this approach is that each application is developed after the previous one, avoiding to define every specific requirement at the beginning of the process. With user stories, epics can be defined at the beginning, and then epics can arise when the development of each application starts. This postponed specification offers more flexibility when changes and issues appear.

Once user stories are defined, they can be added to the backlog: a pile of pending tasks that each user story needs to be completed. This concept is well explained in Methodology (see section 6.1).

Table 3.1 describes the elements of every user story, independently of its level of detail (epic, theme or little user story)

ID	X-NN
Title	-
Description	-
Priority	-
Acc. criteria	-
Parent	-

Table 3.1: User story template.

Next the meaning of each field is described:

- X-NN: is the identifier of the user story. X indicates the category of user story through the following values:
  - E: epic.
  - T: theme.
  - U: little user story.

  NN identifies the user story inside the group of user stories of the same category, adopting a numerical value of two digits. This numerical value will be 01 for first user story of the category and is incremented with the next consecutive number for the next user stories, e.g. E-01, T-02, U-03 (First epic, second theme, and third user story).
- Title: short name of the user story.
- Description: formal definition of the user story, following the format: "As a <user role>, I <want,/need/can/etc...><goal>, so that <reason/benefit/value>". With this predefined clause, each description indicates who is the user implied, which is the intention of the story and what value is obtained with its realization.
- Priority: level of priority of the user story, with the following values: High, Medium, Low.
- Acc. criteria (Acceptance criteria): list with the details of the user story, specifying what is need to consider the user story as done. It also helps to create the depending user stories with a low level of abstraction and define the tests cases for the current user story.
- Parent: identify the user story that has generated the current one, being the parent a higher story in terms of details and abstraction. Obviously, epics do not have any parent since they are the most generic user stories.

3.1 Epics

This section includes the epics definitions.

ID	E-01
Title	**Record dataset**
Description	As a user, I want to record a dataset of gestures, so that I can create a gesture recognizer.
Priority	High
Acc. criteria	• Isolated user interface application • Gestures are recorded with the user gesture performance • Any gesture can be defined • Gestures are stored in a local folder • Dataset is formed by the recorded gestures stored in a local folder • Gesture recording can represent a good or bad example of the gesture
Parent	None

Table 3.2: Epic E-01.

ID	E-02
Title	**Pre-process dataset**
Description	As a user, I want to pre-process a dataset of gestures, so that the dataset is ready to serve as training set for ML algorithms.
Priority	High
Acc. criteria	• User can select the filters to apply • User can decide the input of the filters • User can store the results of the filters • Create a command-line program
Parent	None

Table 3.3: Epic E-02.

ID	E-03
Title	**Train ML algorithm**
Description	As a user, I want to train a ML algorithm with the pre-processed training data, so that I have a gesture recognizer.
Priority	High
Acc. criteria	• Use a machine learning library implementing the machine learning algorithms • User can apply clustering over the dataset • User can apply classifiers over the dataset • User can decide the input dataset • User can store the results of training • Create a command-line program
Parent	None

Table 3.4: Epic E-03.

3.2 Themes

This section includes the themes definitions.

ID	T-01
Title	**User Interface**
Description	As a user, I want to use a user interface application to record a dataset of gestures, so that the dataset recording is intuitive.
Priority	High
Acc. criteria	• Create a WPF application from a Kinect for Windows sample • Give feedback of the state of the recording process • User cannot start recording the gesture until the gesture is selected • User can see a model of its body joints miming the real-life performance • Each gesture is recorded individually
Parent	E-01

Table 3.5: Theme T-01.

ID	T-02
Title	**Gesture name**
Description	As a user, I want to define the name of the gesture, so that I can train personalized gestures.
Priority	Medium
Acc. criteria	• The name can be written by the user • The name can be selected by a gesture list • The name can be stored in a gesture list • User cannot edit the gesture name while Kinect is aware • Kinect is aware when it is ready to start recording, so the user can start recording with a control gesture • Kinect is unaware when it not ready to start recording, so any control gesture of the user can trigger the recording
Parent	E-01

Table 3.6: Theme T-02.

ID	T-03
Title	**Saving path**
Description	As a user, I can select the saving path of the dataset, so that I have control of its location.
Priority	Low
Acc. criteria	• The path can be selected with a window dialog, avoiding the use of non-existing paths • The selected path has to be visible in the user interface • User cannot edit the saving path while Kinect is aware
Parent	E-01

Table 3.7: Theme T-03.

ID	T-04
Title	**Gesture class**
Description	As a user, I want to select the class of the gesture that is going to be recorded, so that I can define its class.
Priority	High
Acc. criteria	• The class must be 'Good' or 'Bad' • 'Good' class will be selected by default • User cannot edit the gesture class while Kinect is aware
Parent	E-01

Table 3.8: Theme T-04.

ID	T-05
Title	**Pre-process dataset application**
Description	As a developer, I need to implement a command-line program, so that users can pre-process a dataset and automate experimentations.
Priority	High
Acc. criteria	• Use Python to develop a dedicated application
Parent	E-02

Table 3.9: Theme T-05.

ID	T-06
Title	**Select filters**
Description	As a user, I want to select the filters to apply over the dataset, so that I can experiment with several combinations of pre-process.
Priority	High
Acc. criteria	• Use a system to read input arguments • The input arguments are filters flags, with a short text identifying the filter, preceded by a "-" and followed by a list of the arguments of each filter (if the filter has arguments)
Parent	E-02

Table 3.10: Theme T-06.

ID	T-07
Title	**Input dataset**
Description	As a user, I want to select which dataset is going to pre-processed, so that I can experiment with any dataset.
Priority	Medium
Acc. criteria	• Users can indicate the files that are part of the dataset • Users can indicate the dataset through one or more folders containing the gestures.
Parent	E-02

Table 3.11: Theme T-07.

ID	T-08
Title	**Pre-process results**
Description	As a user, I want to store the results of the filters, so that the pre-processed dataset can be analyzed and I can decide where the results are stored.
Priority	High
Acc. criteria	• User can indicate the folder where results are going to be stored • Results must be stored in CSV format, but with information regarding the arguments of the pre-process execution that have generated these results
Parent	E-02

Table 3.12: Theme T-08.

ID	T-09
Title	**Apply filters**
Description	As a user, I want to apply transformation filters over a dataset, so that the dataset is pre-processed.
Priority	High
Acc. criteria	• The available filters to apply are: – Normalization and center translation – Filter by feature – Remove frames with lasso gesture – Filter by joint, selecting the joints to be ignored for every gesture – Filter by joint, selecting the joints to be included in every gesture – Unify gestures to a common length, using real frame values – Unify gestures to a common length, using average values • The filters arguments are introduced as program arguments
Parent	E-02

Table 3.13: Theme T-09.

ID	T-10
Title	**Train dataset application**
Description	As a developer, I need to implement a command-line program, so that users can train a dataset and automate experimentations.
Priority	High
Acc. criteria	• Use Python to develop a dedicated application
Parent	E-03

Table 3.14: Theme T-10.

ID	T-11
Title	**Clustering**
Description	As a user, I want to use clustering over the dataset, so that I can experiment with grouped values.
Priority	High
Acc. criteria	• The clustering algorithm to apply is K Means • K Means has to be applied using Scikit-learn library • User can define the number of clusters • User can store the clustering results
Parent	E-03

Table 3.15: Theme T-11.

ID	T-12
Title	**Classifiers**
Description	As a user, I want to train the dataset with different classifiers, so that I can generate a gesture recognizer and compare between models.
Priority	High
Acc. criteria	• The classification algorithms to apply are SVM and KNN • SVM and KNN hast to be applied using Scikit-learn library • User can define training parameters for each classification algorithm
Parent	E-03

Table 3.16: Theme T-12.

ID	T-13
Title	**Input dataset**
Description	As a user, I want to select the input dataset, so that I have control of the experimentation.
Priority	Medium
Acc. criteria	• User can indicate the folders that are part of the dataset
Parent	E-03

Table 3.17: Theme T-13.

ID	T-14
Title	**Training results**
Description	As a user, I want to store the results of the training, so that I complete the experimentation and select a gesture recognizer.
Priority	High
Acc. criteria	• User can indicate the folder where train results are going to be stored • Results are generated by the train performance of the classification algorithm used in the execution
Parent	E-03

Table 3.18: Theme T-14.

ID	T-15
Title	**Select algorithms**
Description	As a user, I want to select an algorithm to apply over the dataset, so that I have control of the experimentation.
Priority	Medium
Acc. criteria	• Use a system to read input arguments • The input arguments are flag parameters of the algorithms, with a short text identifying the parameter of the algorithm, preceded by a "-" and followed by a list of the values required for that parameter • The program will not execute if there are parameters of different algorithms at the same time, giving a feedback message
Parent	E-03

Table 3.19: Theme T-15.

3.3 User stories

This section includes the user stories definitions.

ID	U-01
Title	**User Interface application**
Description	As a developer, I need to implement a user interface application to record a dataset of gestures, so that the user can record a dataset intuitively
Priority	High
Acc. criteria	• Create a WPF application analyzing Kinect for Windows samples • Copy body tracking functionality from BodyBasics-WPF, removing everything not related with tracking functionality • Copy user interface window from ColorBasics-WPF, adapting to mockup (TODO: add reference to mockup)
Parent	T-01

Table 3.20: User story U-01.

ID	U-02
Title	**Play button**
Description	As a user, I want to press a "Play" button, so that Kinect is aware that I want to record a gesture
Priority	High
Acc. criteria	• User cannot start recording the gesture until the gesture is selected • When Play button is pressed the Kinect status changes to aware • When Play button is pressed, the Play button is hidden and the Stop button is visible
Parent	T-01

Table 3.21: User story U-02.

ID	U-03
Title	**Stop button**
Description	As a user, I want to press a "Stop" button, so that Kinect does not have to record any gesture
Priority	High
Acc. criteria	• When Stop button is pressed the Kinect status changes to unaware • When Stop button is pressed, the Stop button is hidden and the Play button is visible
Parent	T-01

Table 3.22: User story U-03.

ID	U-04
Title	**Start with hand lasso**
Description	As a user, I want to start recording a gesture making a lasso gesture with both hands at the same time, so that I can control when the gesture recording starts, without moving from camera vision
Priority	High
Acc. criteria	• When the lasso gesture with both hands at the same time is performed, Kinect status changes to recording
Parent	T-01

Table 3.23: User story U-04.

ID	U-05
Title	**Finish with hand close**
Description	As a user, I want to stop recording a gesture through closing both hands at the same time, so that I can control when the gesture recording finishes, without moving from camera vision
Priority	High
Acc. criteria	• When the close gesture with both hands at the same time is performed, Kinect status changes to aware • If the user delays more than 30 seconds, gesture recording will stop.
Parent	T-01

Table 3.24: User story U-05.

ID	U-06
Title	**Feedback**
Description	As a user, I want to receive feedback of the system telling me if I am recording or not, so that I know if the gesture recording is working fine or not
Priority	High
Acc. criteria	• Text messages appear in the top of the window • An information button appears in the position of Play/Stop button when no gesture is selected, indicating that the user must select a gesture • Time duration in seconds appears at the top right of the window, indicating the number of seconds that the current gesture is lasting or has been lasted (in case the recording finished). At the right side of this value there is the amount of frames. • Amount of frames appears at the top right of the window, indicating the number of frames that the current gesture is lasting or has been lasted (in case the recording finished). At the left side of this value there is the amount of frames. • A text must indicate the name of the gesture that is ready to be recorded, with an explanatory text at the top gesture name.
Parent	T-01

Table 3.25: User story U-06.

ID	U-07
Title	**Real time representation**
Description	As a user, I want to see a real-time representation of my body when I am using the user interface application, so that I know what is my position regarding the camera
Priority	Medium
Acc. criteria	• This functionality is inherited from BodyBasics-WPF
Parent	T-01

Table 3.26: User story U-07.

ID	U-08
Title	**Gesture information**
Description	As a user, I want to record all the information available of a gesture, so that I have all this information available for pre-processing, learning and experimenting
Priority	High
Acc. criteria	• The gesture information is stored in one single CSV file. • Gesture information: name of the gesture, class of the gesture, number of frames that the tracking process has needed, number of seconds that the tracking process has needed, maximum duration (in seconds) allowed for the tracking process, control value to check if the tracking process has finished because it was manually stopped or because the time deadline was reached, and information of every frame. • Frame information: frame identifier, second when frame was recorded, left hand confidence, left hand state, right hand confidence, right hand state, value indicating if the gesture is restricted in that frame, number of joints of the tracked body, lean vector of the body in that frame, tracking state of body lean model, auto-generated identifier of the tracked body, location of the window edge that is clipped, and information of every joint • Joint information: type of joint, tracking state of the joint, depth space position vector, camera space position vector, orientation vector • Each file has information of the parameters of the execution that has generated this file.
Parent	T-01

Table 3.27: User story U-08.

ID	U-09
Title	**Gesture information format**
Description	As a user, I want to select the level of detail of the gesture data that is going to be stored, so that I can have data with or without labels that describes the values of the gestures
Priority	High
Acc. criteria	• Selecting the use of labels in data has to be implemented with a checkbox with the name "Raw Data", checked by default. • Checked value indicates that the data is going to be stored without labels. • Non-checked indicates that the data is going to be stored with labels, adding a header with the label information at the top of each gesture file.
Parent	T-01

Table 3.28: User story U-09.

ID	U-10
Title	**Gesture name typed**
Description	As a user, I can write the name of the gesture by typing it, so that I can define new personalized gestures
Priority	Medium
Acc. criteria	• Use an spell checking • An explanatory text of the field appears at the top of the gesture name • The gesture name must be implemented as a text box
Parent	T-02

Table 3.29: User story U-10.

ID	U-11
Title	**Gesture name selected**
Description	As a user, I can select the name of the gesture from a list of existing ones, so that I can choose a previous defined gesture
Priority	Medium
Acc. criteria	• Use a predefined hardcoded list of common gestures, involving hands and/or arms • An explanatory text of the field appears at the top of the list • The list must be implemented as a combo box
Parent	T-02

Table 3.30: User story U-11.

ID	U-12
Title	**Adding gesture name to list**
Description	As a user, I can introduce a new gesture in a list with existing gestures, so that the new gesture does not need to be defined again
Priority	Low
Acc. criteria	• The content of the list must persist between different executions of the application.
Parent	T-02

Table 3.31: User story U-12.

ID	U-13
Title	**Change current gesture**
Description	As a user, I want to change the current gesture that is going to be recorded, so that I can record different gestures in the same application execution
Priority	Medium
Acc. criteria	• The change of the current gesture can be made from the written gesture or the picked from the gesture list
Parent	T-02

Table 3.32: User story U-13.

ID	U-14
Title	**Saving path dialog**
Description	As a user, I can select the saving path through a window dialog, so that I have control of a real location
Priority	Low
Acc. criteria	• Default path: Windows system desktop • An explanatory text of the field appears at the top of the path text
Parent	T-03

Table 3.33: User story U-14.

ID	U-15
Title	**Saving path feedback**
Description	As a user, I can see the selected saving path, so that I know where the dataset is going to be stored
Priority	Low
Acc. criteria	-
Parent	T-03

Table 3.34: User story U-15.

ID	U-16
Title	**Class values**
Description	As a user, I want to select if the recording is going to be a 'Good' or 'Bad' representation of the gesture, so that I can define its class
Priority	High
Acc. criteria	• The class selection must be implemented with radio buttons. • An explanatory text of the field appears at the top of the radio buttons
Parent	T-04

Table 3.35: User story U-16.

ID	U-17
Title	**Good value by default**
Description	As a user, I do not want to select the 'Good' class every time I start the application, so that I avoid a repetitive task
Priority	Medium
Acc. criteria	-
Parent	T-04

Table 3.36: User story U-17.

ID	U-18
Title	**Pre-process dataset application in Python**
Description	As a developer, I need to implement a command-line program in Python, so that users can pre-process a dataset and automate experimentations
Priority	High
Acc. criteria	-
Parent	T-05

Table 3.37: User story U-18.

ID	U-19
Title	**Filter flag**
Description	As a user, I want to select the filter using a flag argument, so that I can automate experimentations with a script
Priority	Medium
Acc. criteria	• Each flag argument is preceded by a "-" character and followed by the list of the filter arguments (if the filter has arguments) • Program will not execute if the user introduces a non-existing flag, giving a feedback message
Parent	T-06

Table 3.38: User story U-19.

ID	U-20
Title	**Filter flag arguments**
Description	As a user, I want to select each filter argument, so that I can experiment with each filter
Priority	Medium
Acc. criteria	• Arguments are introduced after the filter flag, separated by spaces • If the user introduces arguments for a filter that does not have arguments, the program will not execute, giving a feedback message
Parent	T-06

Table 3.39: User story U-20.

ID	U-21
Title	**Select input files**
Description	As a user, I want to select the files to pre-process, so that I can experiment with single or few gestures
Priority	Low
Acc. criteria	• User has to indicate this type of input with a flag argument • The flag argument must be followed by a list of paths of files, containing gestures definitions • Program will not execute if any path does not exist, giving a feedback message
Parent	T-07

Table 3.40: User story U-21.

ID	U-22
Title	**Select input folders**
Description	As a user, I want to select the folders to pre-process, so that I can experiment with entire datasets
Priority	Medium
Acc. criteria	• User has to indicate this type of input with a flag argument • The flag argument must be followed by a list of paths of folders, containing files with the gestures definitions inside • Program will not execute if any path does not exist, giving a feedback message
Parent	T-07

Table 3.41: User story U-22.

ID	U-23
Title	**Select output folder**
Description	As a user, I want to select the output folder where the results of the filters are going to be stored, so that I have control of the pre-processing result
Priority	Medium
Acc. criteria	• User has to indicate the output folder with a flag argument • The flag argument must be followed by the path of the folder • Program will not execute if the path does not exist, giving a feedback message
Parent	T-08

Table 3.42: User story U-23.

ID	U-24
Title	**Results**
Description	As a user, I want to store the results of the filters in CSV format, so that I can train a processed dataset
Priority	High
Acc. criteria	• The results have a header with information regarding the arguments of the pre-process execution that have generated these results • The results are new files, where each file is the processed version of the original one that has been introduced in the program as an input • The file names are the same that the original ones, but having a "processed.csv" label at the end
Parent	T-08

Table 3.43: User story U-24.

ID	U-25
Title	**Center translation filter**
Description	As a user, I can translate the positions of each gesture to the center of the camera, so that the positions of all gestures are comparable
Priority	High
Acc. criteria	• Implement normalization filter, being executed in first place • No arguments • Follow the formulas defined in *Easy gesture recognition for Kinect paper* [8]
Parent	T-09

Table 3.44: User story U-25.

ID	U-26
Title	**Normalization filter**
Description	As a user, I can normalize the positions of each gesture to a common body size, so that the sizes of all gestures are comparable
Priority	High
Acc. criteria	• Implement with Center translation filter, being executed in second place • No arguments • Follow the formulas defined in *Easy gesture recognition for Kinect paper* [8]
Parent	T-09

Table 3.45: User story U-26.

ID	U-27
Title	**Features filter**
Description	As a user, I can filter any feature for all the gestures, so that non-relevant features are ignored
Priority	High
Acc. criteria	• Arguments: list with the features names • Show a help text in case of the user introduces a wrong feature name, indicating all the possible values
Parent	T-09

Table 3.46: User story U-27.

ID	U-28
Title	**Lasso filter**
Description	As a user, I can filter the frames where both hands are performing lasso gesture at the same time, for all gestures, so that non-relevant frames are ignored
Priority	High
Acc. criteria	• Start removing frames from the beginning of each gesture and stops checking when the lasso condition does not match. • No arguments
Parent	T-09

Table 3.47: User story U-28.

ID	U-29
Title	**Exclude joint filter**
Description	As a user, I can select any joint to not be included for all the gestures, so that non-relevant joints are ignored
Priority	High
Acc. criteria	• Arguments: list with the joints names • Show a help text in case of the user introduces a wrong joint name, indicating all the possible values • Maintain the joints that has to be used in Center translation and Normalization filters, until the end of the execution. • Program will not execute if both Exclude and Include joint filters are selected, giving a feedback message
Parent	T-09

Table 3.48: User story U-29.

ID	U-30
Title	**Include joint filter**
Description	As a user, I can select any joint to be included for all the gestures, so that I choose the relevant joints
Priority	High
Acc. criteria	• Arguments: list with the joints names • Show a help text in case of the user introduces a wrong joint name, indicating all the possible values • Maintain the joints that has to be used in Center translation and Normalization filters, until the end of the execution. • Program will not execute if both Exclude and Include joint filters are selected, giving a feedback message
Parent	T-09

Table 3.49: User story U-30.

ID	U-31
Title	**Same length with original values filter**
Description	As a user, I can change the length of all the gestures to a common length, where frames for each gesture are picked through a certain frequency, so that all gesture have the same length and its original values
Priority	High
Acc. criteria	• Arguments: positive numerical value indicating the target size • Program will not execute if both Same length filters are selected, giving a feedback message • Show a help text in case of the user introduces a numerical value, indicating that a positive numerical value has to be introduced
Parent	T-09

Table 3.50: User story U-31.

ID	U-32
Title	**Same length with average values filter**
Description	As a user, I can change the length of all the gestures to a common length, where frames for each gesture are averaged through a certain frequency, so that all gesture have the same length and average values
Priority	High
Acc. criteria	• Arguments: positive numerical value indicating the target size • Program will not execute if both Same length filters are selected, giving a feedback message • Show a help text in case of the user introduces a numerical value, indicating that a positive numerical value has to be introduced
Parent	T-09

Table 3.51: User story U-32.

ID	U-33
Title	**Train dataset application in Python**
Description	As a developer, I need to implement a command-line program in Python, so that users can train a dataset and automate experimentations
Priority	High
Acc. criteria	-
Parent	T-10

Table 3.52: User story U-33.

ID	U-34
Title	**K Means**
Description	As a user, I want to use clustering over the dataset using K Means, so that I can experiment with grouped values
Priority	High
Acc. criteria	• K Means has to be applied using Scikit-learn library • Clustering is applied before training the dataset • Arguments: positive numerical value indicating the number of clusters
Parent	T-11

Table 3.53: User story U-34.

ID	U-35
Title	**Clustering results**
Description	As a user, I want to store the clustering results, so that I can analyze the clustering experimentation
Priority	Medium
Acc. criteria	• Clustering results to store: – Clusters centroids – Distances between gestures and centroids – Predicted cluster for each gesture – Score based on distances – Graphics representing the gestures grouped by clusters, when gesture data is bi-dimensional
Parent	T-11

Table 3.54: User story U-35.

ID	U-36
Title	**SVM**
Description	As a user, I want to train the dataset using SVM, so that I can generate a gesture recognizer
Priority	High
Acc. criteria	• SVM has to be applied using Scikit-learn library • Program will not execute if KNN is also selected, giving a feedback message • Arguments: – Type of SVM: c, nu or linear – Type of Kernel: linear, poly, rbf, sigmoid or pre-computed – Training dataset distribution in percentages for the sub-datasets: traning, test and validation (optional) • Arguments are not mandatory
Parent	T-12

Table 3.55: User story U-36.

ID	U-37
Title	**KNN**
Description	As a user, I want to train the dataset using KNN, so that I can generate a gesture recognizer
Priority	High
Acc. criteria	• KNN has to be applied using Scikit-learn library • Program will not execute if SVM is also selected, giving a feedback message • Arguments: – Number of neighbors (numerical positive value) – Training dataset distribution in percentages for the sub-datasets: traning, test and validation (optional) • Arguments are not mandatory
Parent	T-12

Table 3.56: User story U-37.

ID	U-38
Title	**Select input folder**
Description	As a user, I want to select the folder of the dataset to be trained, so that I can experiment with any dataset
Priority	Medium
Acc. criteria	• User has to indicate the input folder with a flag argument • The flag argument must be followed by the paths of the folder, containing files with the gestures definitions inside • Program will not execute if the path does not exist, giving a feedback message
Parent	T-13

Table 3.57: User story U-38.

ID	U-39
Title	**Select output folder**
Description	As a user, I want to select the output folder where the results of the training are going to be stored, so that I have control of the training experimentation
Priority	Medium
Acc. criteria	• User has to indicate the output folder with a flag argument • The flag argument must be followed by the path of the folder • Program will not execute if the path does not exist, giving a feedback message
Parent	T-14

Table 3.58: User story U-39.

ID	U-40
Title	**SVM results**
Description	As a user, I want to store the results of the training using SVM, so that I complete the experimentation of SVM
Priority	High
Acc. criteria	• SVM results to store: – Classification scores over each of the sub-datasets: training, test and validation (optional) – Classification score over the complete dataset – Confusion matrix over the complete dataset – Cross-validation score over the complete dataset – Leave-one-out score over the complete dataset – SVM parameters used in the execution
Parent	T-14

Table 3.59: User story U-40.

ID	U-41
Title	**KNN results**
Description	As a user, I want to store the results of the training using KNN, so that I complete the experimentation of KNN
Priority	High
Acc. criteria	• KNN results to store: – Classification scores over each of the sub-datasets: training, test and validation (optional) – Classification score over the complete dataset – Confusion matrix over the complete dataset – Cross-validation score over the complete dataset – Leave-one-out score over the complete dataset – KNN parameters used in the execution
Parent	T-14

Table 3.60: User story U-41.

ID	U-42
Title	**Select algorithm flag**
Description	As a user, I want to select an algorithm to apply over the dataset, using a flag argument, so that I have control of the experimentation
Priority	Medium
Acc. criteria	• Each algorithm is automatically selected when one of its parameters appears • The program will not execute if there are parameters of different algorithms at the same time, giving a feedback message
Parent	T-15

Table 3.61: User story U-42.

ID	U-43
Title	**Select parameter algorithm flag**
Description	As a user, I want to select any parameter of an algorithm, using a flag argument, so that I can experiment with each algorithm
Priority	Medium
Acc. criteria	• A parameter is defined as a short text identifying the parameter of the algorithm, preceded by a "-" and followed by a list of the values required for that parameter (if requires any)
Parent	T-15

Table 3.62: User story U-43.

Chapter 4

Development

This section gives an overview of the solutions of the project. It also resumes the development environment and the interaction between the applications.

In order to afford the task of creating an application to support the creation of gesture recognition models (also called recognizers), the functionality has been distributed between three applications:

- An application in WPF to record personalized gestures.
- An application in Python to pre-process the recorded data.
- An application in Python to train machine learning algorithms to generate gesture recognition models using the pre-processed data.

Then one section is dedicated for every application, detailing its design, implementation, and experimentation.

4.1 Development environment

Hardware requirements The WPF application to record the gestures uses the Kinect v2 sensor and an adapter to communicate with a computer: Kinect Adapter for Windows. Since the application uses Kinect for Windows SDK 2.0, the computer where this application was developed follows the Microsoft system requirements recommendations. The conditions fulfilled by the computer are:

- Windows 8.1 (x64).
- 64 bit (x64) processor.
- 8 GB Memory.
- I7 2.30 GHz, despite I7 3.1 GHz or higher is required.
- Built-in USB 3.0 host controller (Intel chipset).
- DX11 capable graphics adapter: GeForce 820M, despite is not indicated in the recommended list.

Software requirements The following software was installed to develop the WPF application:

- Visual Studio 2015 Community version, despite Microsoft recommends hat Kinect applications should be developed by Visual Studio 2012 or Visual Studio 2013.
- Microsoft .NET framework 4.6 SDK.
- Kinect for Windows SDK v2.0.

In the case of Python applications, the required software was:

- Python 3.6.3.
- Scikit-learn library: to apply ML algorithms like classification and clustering.
- NumPy library: to create data structures.

4.2 Interaction between applications

Though WPF, pre-process and machine learning applications are isolated, Figure 4.1 shows a diagram of the inputs and outputs of each of them.

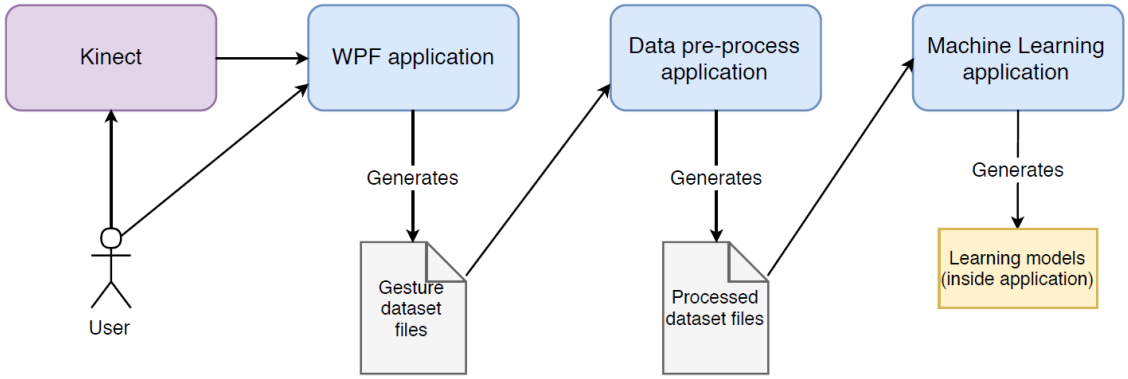

Figure 4.1: Interaction between applications.

To summarize:

- WPF application generates dataset files that are consumed by the pre-process application. The data is recorded by the Kinect sensor and configured with the parameters introduced by the user in the WPF interface.
- Data pre-process application process the dataset and generate dataset files that are consumed by the machine learning application.
- Machine learning application train classification algorithms and generate learning models that are maintained inside the application.

4.3 Recording application

The first step to generate gesture recognition models is to generate the training dataset. A dataset is an aggregate of data of the same type, in this case, human gestures. Microsoft Kinect v2 is the sensor used in this project to generate the dataset, because the sensor is able to track the positions and orientations of body joints in real-time, among other body features.

I have developed a WPF application based on a UI that allows users to record any type of gesture. The application was named *Kinect Gesture Reader*.

4.3.1 Functionality and guide of use

The main functionality of Kinect Gesture Reader is to record personalized gestures defined by the users. This is possible because the Kinect tracks joints of the full body and users only have to perform the gestures by themselves in front of the camera.

In addition to this, users can manage some features of the gestures before the recording process starts:

- Name of the gesture.
- Class of the gesture. There are two possible classes, a positive class, representing a good example of the gesture, and a negative class, representing an example of a different gesture which is a bad example of the gesture to recognize.
- The path where the recorded gesture is stored.
- Gesture list, where users can add gestures names so they can be used in the future.
- The format of the stored dataset (see subsection 4.3.3).

The Figure 4.2 shows the UI application with numbers indicating its components.

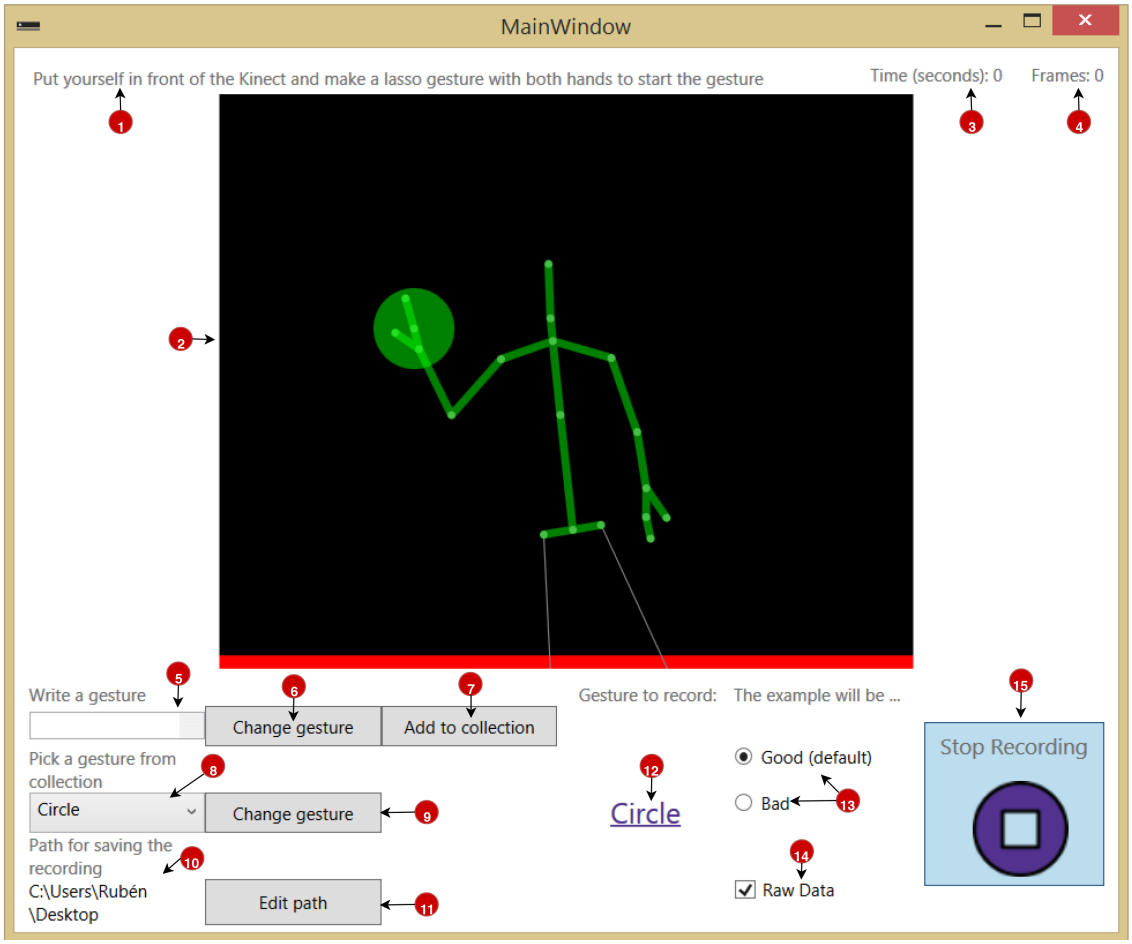

Figure 4.2: WPF application with information points.

Here there is the explanation that match with the number points of Figure 4.2:

- 1: Informative text about the state of the system: gives feedback of the Kinect availability and the recording process.
- 2: Screen with a real-time model of the tracked body.
- 3: Informative text about the number of seconds that the gesture recording is lasting.
- 4: Informative text about the number of frames that the gesture recording is lasting.
- 5: Text box to write the name of the gesture.
- 6: Button to change the gesture that can be recorded with the typed name in the Text box.
- 7: Button to add the gesture name to the list of gestures.
- 8: Combo box with the list of stored gestures.
- 9: Button to change the gesture that can be recorded with the selected in the Combo box.

- 10: Path where the gesture can be stored.
- 11: Button to change the path where the gesture can be stored.
- 12: Name of the gesture that can be recorded.
- 13: Radio buttons to select the class of the gesture that can be recorded.
- 14: Checkbox to indicate in which format the gesture can be recorded.
- 15: Play/Stop button to start and end the gesture recording.

At the beginning of the application execution, there is no gesture selected to be recorded, and the Play/Stop button has the aspect shown in Figure 4.3. This figure also shows the scenario where Kinect sensor is not connected. The gesture recording cannot start if no gesture is selected, so the user has to select it by typing it or selecting it from the gesture list. Figure 4.2 shows the application when one gesture has been selected and the Play button has been pressed, so the recording can start.

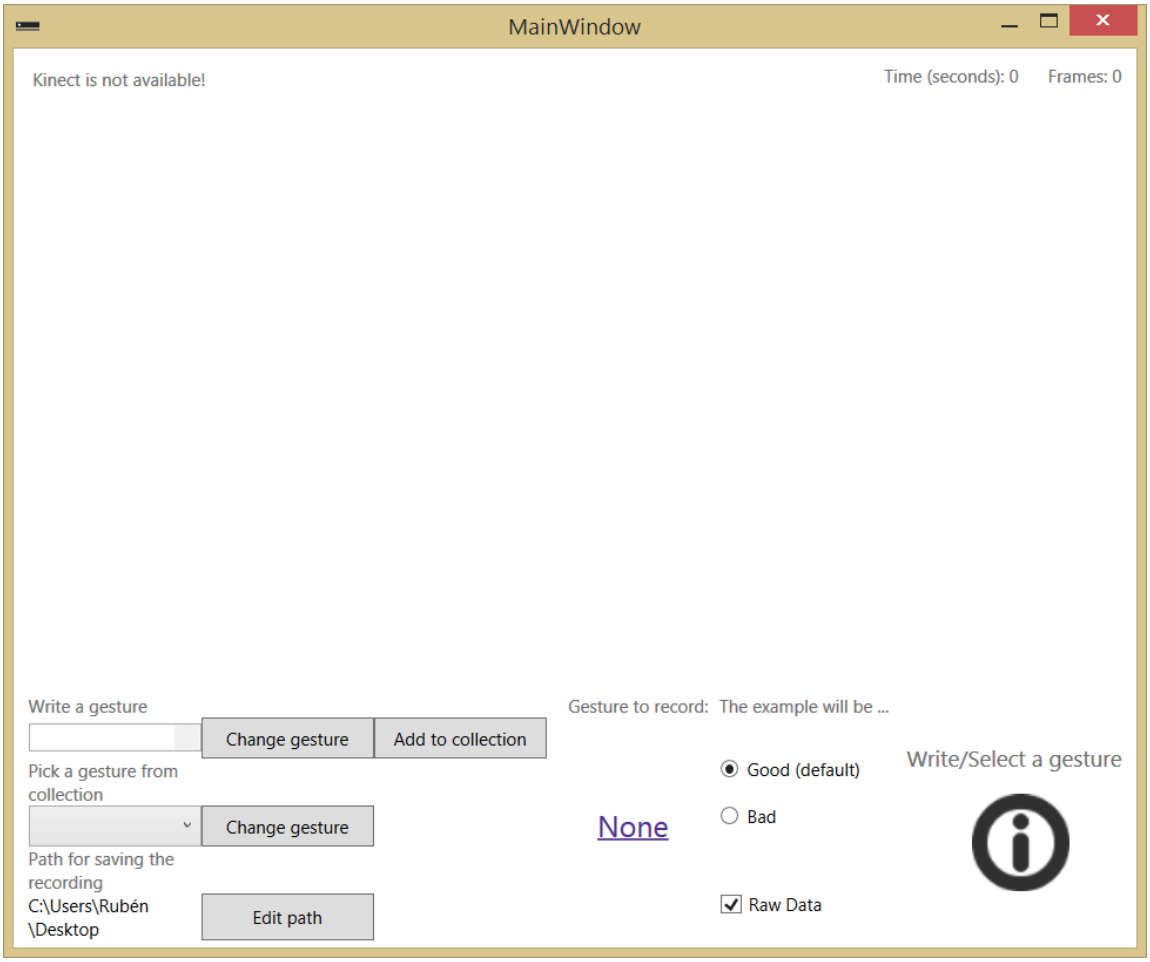

Figure 4.3: WPF application at the beginning of the execution.

The application differentiates between four states of the Kinect sensor:

- Non-available: it may be not connected or not available because of other reason.
- Is aware: the Kinect sensor is able to start recording a gesture. The sensor can react only to the "Start control gesture", a gesture to indicate that the recording must start.
- Is unaware: the Kinect sensor is not able to record. The sensor receives information but is discarded.
- Is recording: the Kinect sensor is recording a gesture. The sensor receives information and stores it.

The main reason for using this classification is because the application is designed to be used by a single person. A way of achieving this functionality is by the use of control gestures. Control gestures allow the control of the recording process without having to click the buttons of the UI, so the user that performs the gesture is placed in the right position since the first moment. When a user selects a gesture, he/she can press the Play button and place in front of the camera and perform the 'Start control gesture': a lasso gesture with both hands at the same time (Figure 4.4).

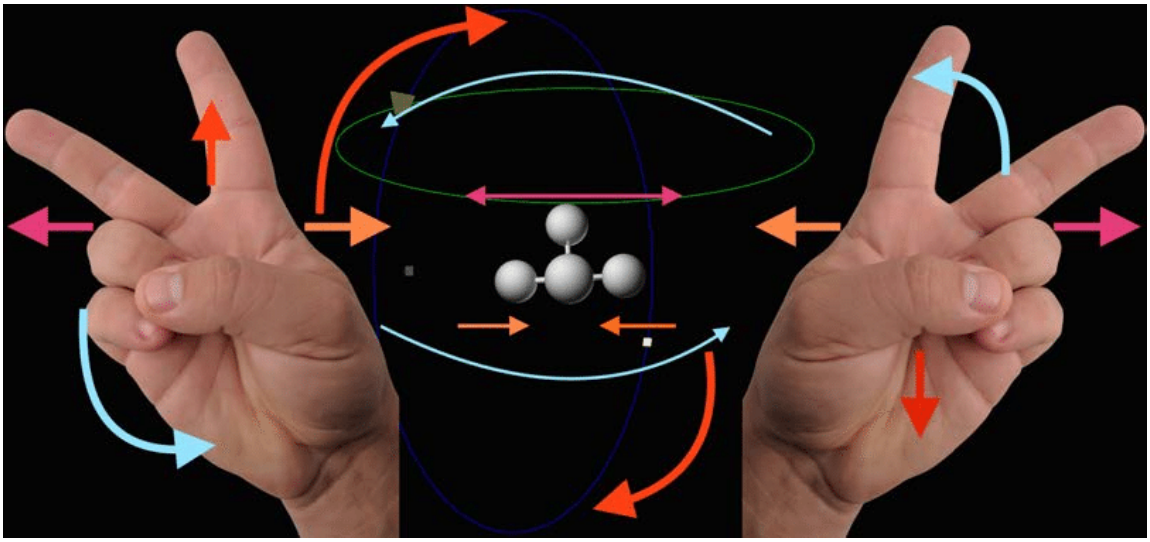

Figure 4.4: Lasso gesture[61].

When this event occurs the Kinect starts recording the actions performed by the user, that can finalize when the user performs the 'Stop control gesture': both hands closed at the same time. Then the recorded gesture is stored in the saving path defined previously (Windows system Desktop by default).

Noise data However, there is a drawback with this control system. The first moments of the gesture have both hands in lasso position, adding noise to the data. This issue has been considered and a transformation algorithm for reducing that noise is implemented in the pre-process application (see section 4.4).

Unexpected behaviors There are some scenarios that can affect the normal behavior of the system. They are described in the following points with the appropriate action applied.

- In order to prevent the application taking much memory from the computer, a limit of 30 seconds, has been designated as the maximum time that any gesture recording can last. This is considered as an ample time to perform even long gestures. If this maximum time is reached, the recording process is interrupted by a 'Time deadline event'.
- Another scenario that was taken into account was a user trying to change the gesture data (name, class, saving path, format) when the Kinect sensor is aware or recording. This action has been restricted so data corruption or undesirable behavior is avoided, though this application is designed for one user, there is the possibility that a second person could modify the data while the first one is in front of the camera.
- When the user disappears while the recording process has not finished (neither 'Stop control gesture' nor 'Time deadline event'), the recording process is interrupted, the gesture data is removed and Kinect pass to the unaware state.

Consecutive recordings In order that the gesture recording becomes a fast process, the application allows to start recording a new gesture just after the finalization of the previous one, i.e. the user performs the 'Stop control gesture' to record the gesture so it is stored in a CSV file, and then performs the 'Star control gesture' to initiate a new gesture.

Flow diagram Figures 4.5 helps to visualize all the possible scenarios and the execution flow of the system. Circles represent states of the systems, being the red circle the initial point when the system starts, and the green circle the state when a gesture has been stored in the predefined path. The texts next to each arrow indicate the actions that trigger the transition between states. The green arrow indicates the transition from a gesture being saved and the state where Kinect is aware and the user start recording again. The red arrow indicates the transition from a state where Kinect is recording and the user disappears from the Kinect view, so the recording process is aborted and Kinect is unaware.

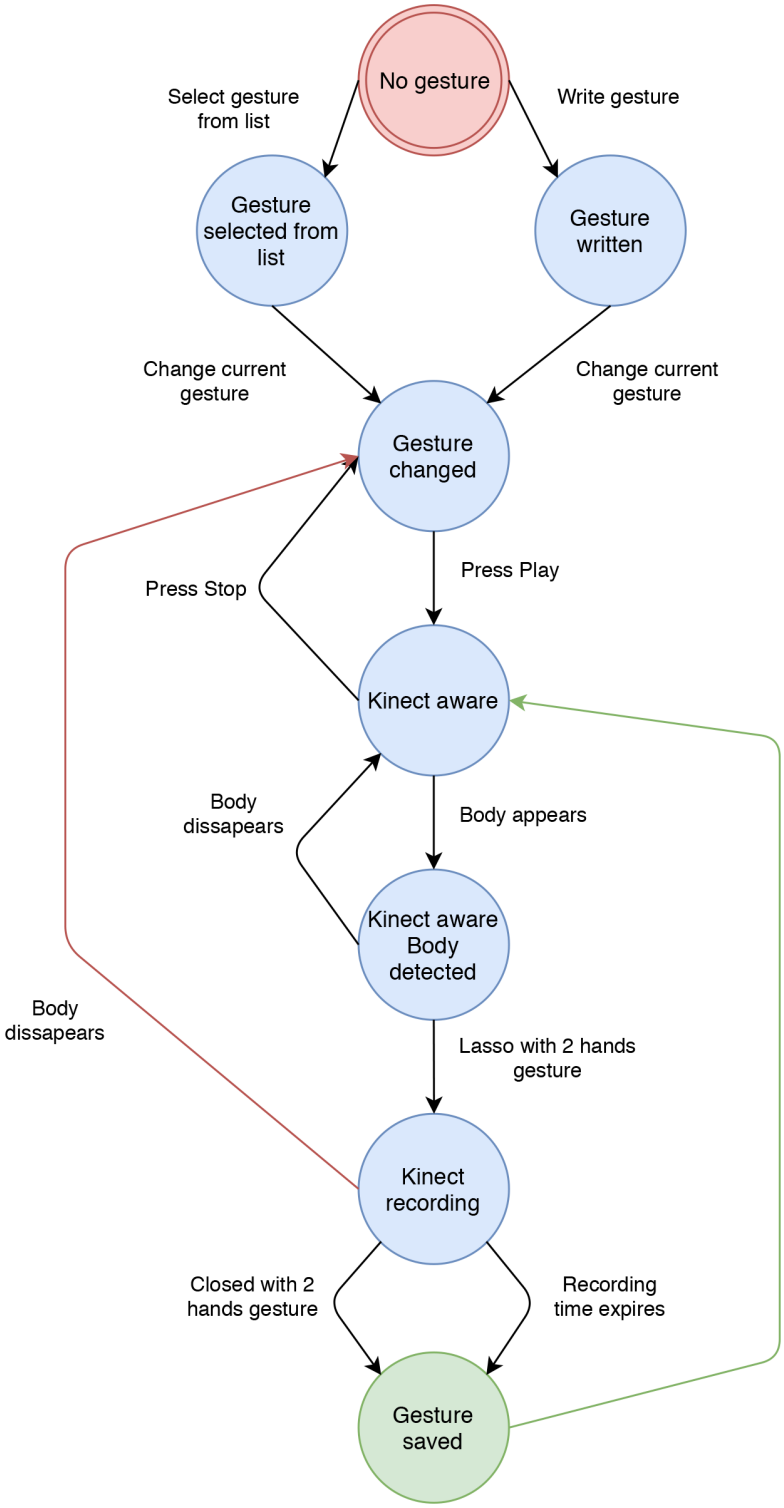

Figure 4.5: WPF application execution flow.

The transition between states is always accompanied with a feedback message update. Tables 4.1 to 4.20 include a list of all the feedback messages used by the application (note that some of this feedback messages are not shown in the recording flow because they give feedback of the gesture management or other issues that the system must afford):

ID	Feed-01
Message	**Kinect lost the body tracking, press Play button to start the gesture again**
Scenario	User disappeared so Play button must be pressed again to start recording a gesture

Table 4.1: Feedback message `Feed-01`.

ID	Feed-02
Message	**Failed to save gesture to <save location>. There was no tracked content**
Scenario	By some reason, tracking buffer had not any value. The tracking buffer is used internally to add the frames while Kinect is recording

Table 4.2: Feedback message `Feed-02`.

ID	Feed-03
Message	**Failed to save gesture to <save location>, error occurred while writing**
Scenario	Gesture is not stored because an error occurred while writing

Table 4.3: Feedback message `Feed-03`.

ID	Feed-04
Message	**You can't manage the gesture configuration while Kinect is aware**
Scenario	Kinect sensor is aware so the user cannot change any data until it is unaware again

Table 4.4: Feedback message `Feed-04`.

ID	Feed-05
Message	**Gesture changed with success**
Scenario	Gesture changed from the list of gestures

Table 4.5: Feedback message `Feed-05`.

ID	Feed-06
Message	**Gesture added to collection with success**
Scenario	Gesture added to the list of gestures

Table 4.6: Feedback message `Feed-06`.

ID	Feed-07
Message	**Gesture changed with success**
Scenario	Gesture changed from the text box

Table 4.7: Feedback message `Feed-07`.

ID	Feed-08
Message	**You must select a gesture first**
Scenario	User tries to change the gesture from the list of gestures but any gesture from that list was selected

Table 4.8: Feedback message `Feed-08`.

ID	Feed-09
Message	**You must write the gesture first**
Scenario	User tries to change the gesture from the text box but any gesture was written

Table 4.9: Feedback message `Feed-09`.

ID	Feed-10
Message	**Path edited with success**
Scenario	Saving path was changed with success

Table 4.10: Feedback message `Feed-10`.

ID	Feed-11
Message	**Repeated gesture in the collection, please add a different one**
Scenario	User tries to add an existing gesture to the gesture list but it is not allowed

Table 4.11: Feedback message `Feed-11`.

ID	Feed-12
Message	**Kinect is running**
Scenario	Kinect sensor is running and unaware

Table 4.12: Feedback message `Feed-12`.

ID	Feed-13
Message	**Saved gesture to <save location> because of the time deadline. Make a lasso gesture with both hands to start a new gesture. Press Stop button to stop recording**
Scenario	Time to record expired so the gesture is saved automatically

Table 4.13: Feedback message `Feed-13`.

ID	Feed-14
Message	**Saved gesture to <save location>. Make a lasso gesture with both hands to start a new gesture. Press Stop button to stop recording**
Scenario	Gesture recorded with success. User made both hands closed gesture to stop recording and save the gesture. The user can start a new gesture or change Kinect to unaware by pressing the Stop button

Table 4.14: Feedback message `Feed-14`.

ID	Feed-15
Message	**Kinect is not available!**
Scenario	Kinect sensor is not available

Table 4.15: Feedback message `Feed-15`.

ID	Feed-16
Message	**Make a lasso gesture with both hands to start the gesture**
Scenario	Kinect is aware and the user is in front of the Kinect. The user has to make a lasso gesture with both hands to start recording, or press Stop button to change Kinect to unaware

Table 4.16: Feedback message `Feed-16`.

ID	Feed-17
Message	**Put yourself in front of the Kinect and make a lasso gesture with both hands to start the gesture**
Scenario	Kinect is aware but user have to be placed in front of the Kinect

Table 4.17: Feedback message `Feed-17`.

ID	Feed-18
Message	**You interrupted the tracking process manually**
Scenario	User is recording a gesture but the Stop button is pressed to stop recording

Table 4.18: Feedback message `Feed-18`.

ID	Feed-19
Message	**You stopped the Kinect recording. Press Start Recording to start a new gesture**
Scenario	User pressed stop button while Kinect is aware. Now the user has to press the Play button so Kinect is aware and can start recording. Now the user can manage the gesture properties too

Table 4.19: Feedback message `Feed-19`.

ID	Feed-20
Message	**Close both hands to finish and save the gesture.Press Stop to interrupt the recording**
Scenario	Kinect is recording a gesture so the user can stop recording the gesture with both hands closed. The gesture would be saved in the CSV file when this is done

Table 4.20: Feedback message `Feed-20`.

4.3.2 Implementation details

Prior to the implementation of Kinect Gesture Reader, the Kinect for Windows SDK 2.0 was analyzed, in order to find reusable projects that could afford the same or similar functionality than the desired for the application. A set of samples of Kinect applications, called SDK Browser v2.0 (Kinect for Windows), is installed with the SDK. This set has Kinect samples that use different frameworks of the Kinect SDK, like DepthFrameReader (access to depth tracked data) or FaceFrameReader (access to tracked characteristics of the face). There was found that the BodyBasics-WPF sample uses the BodyFrameReader to obtain and visualize the body frames captures by the Kinect sensor. In another hand, ColorBasics-WPF sample has a Screenshot button to store images, so it was useful to understand how to store data with WPF. The entire tracking part of the BodyBasics-WPF functionality is reused in Kinect Gesture Reader, leaving more capacity to focus on the UI changes and the recording logic (BodyBasics-WPF tracks the information but it is not stored in any buffer).

4.3.3 Gesture data

As mentioned before, Kinect Gesture Reader communicates with Kinect sensor to store the information that Kinect tracks while the user is performing a gesture. The data tracked in this process is composed by a sequence of frames with information of the human body and general information of the tracked body itself. Each frame contains specific information of the body in that frame, including data from the 25 joints of the human body that Kinect is able to track.

All this information is stored in CSV files, with the following problematic: all the information regarding frames is a sequence of frames with the same information, but the general gesture information only appears at the beginning of the CSV file. So it is not possible to follow a proper definition of a CSV file.

In order to facilitate the processing of this files by the pre-process application (see section 4.4), some control characters (`: % $ #`) have been placed between the general gesture information and frames information, and even between frames information.

Figure 4.6 shows an overview of the positions of the 25 joints tracked by the Kinect v2.

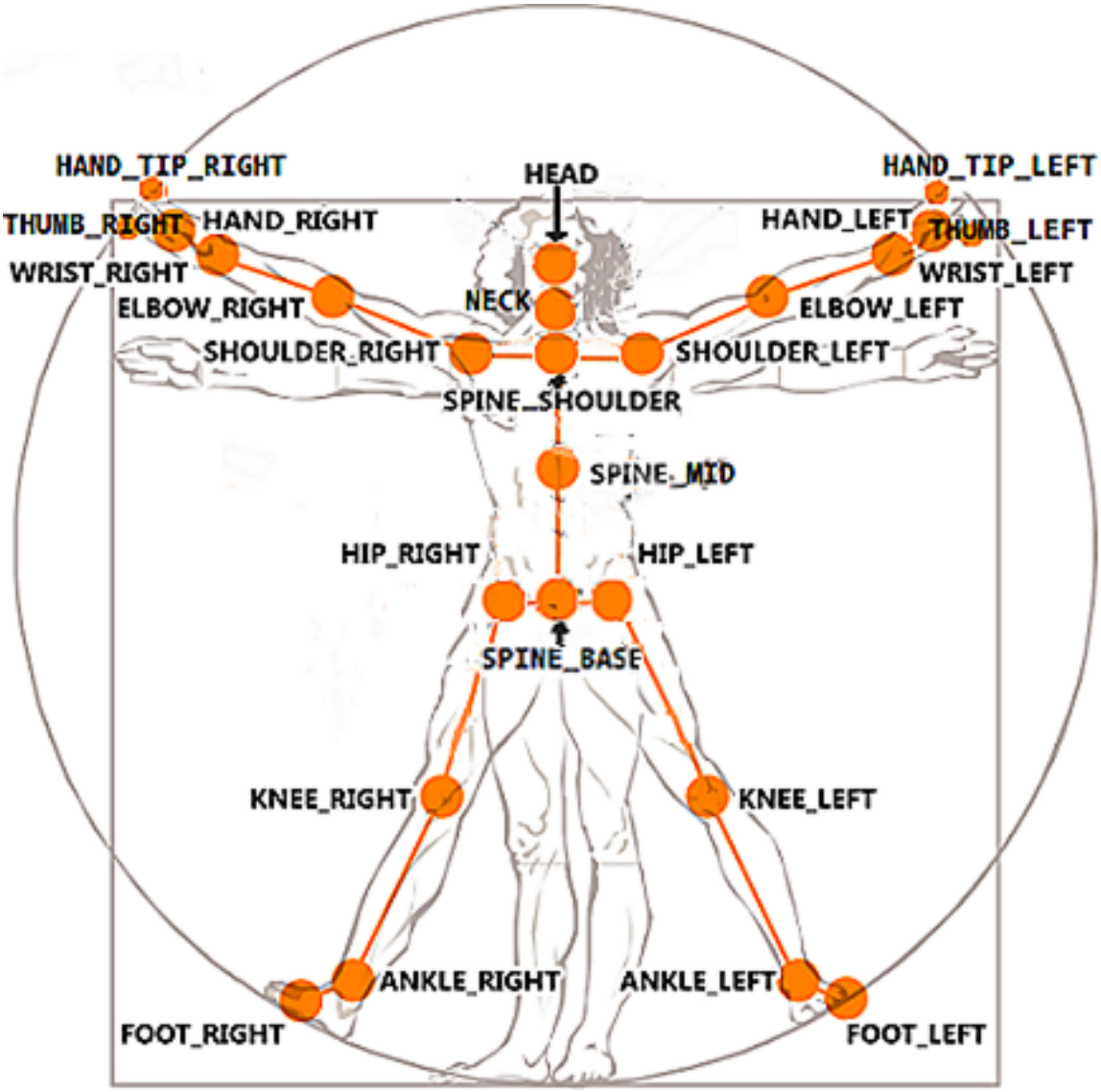

Figure 4.6: Skeleton positions relative to the human body [62].

The list of all the features that are recorded for every gesture is the following [63]:

General gesture data

- Name of the gesture.
- Class of the gesture.
- Number of frames of the gesture.
- Number of seconds that the gesture recording lasted.
- Maximum duration (in seconds) allowed for the recording.

- Control value to check if the recording process finished because of the 'Stop control gesture' or the 'Time deadline event'.

Frame data for every recorded frame of the gesture

- Identifier of the frame in the sequence of frames that conform the gesture.
- Time (in seconds) when the frame was recorded.
- Left-hand confidence. Possible values: High (hand is tracked), Low (hand is not tracked). When Kinect states that some joint is not tracked it means that it is not visible for the sensor.
- Left-hand state. Possible values: Closed, Lasso, NotTracked, Open and Unknown (used when the hand is not making any gesture but is tracked).
- Right-hand confidence. Same values as the left-hand confidence.
- Right-hand state. Same values as the left-hand state.
- Is Restricted: indicates if the tracked information is complete or partially tracked because of the application that tracks this information.
- Number of joints of the tracked body. In this case, Kinect sensor counts joints even if they are inferred and not tracked. Inferred is a tracking state of the joints when they are not visible but its information can be inferred by the data of the visible joints.
- Lean vector of the body, having coordinates X and Y. It indicates in which grade the body is leaning right, left, back or front.
- Tracking state of the body. Possible values: Inferred, NotTracked, Tracked.
- Identifier of the tracked body randomly generated when the body appears in front of the Kinect.
- Location of the window edge that is clipped. This is the edge where the body is interrupted and do not filled in the window space. Possible values: None (any edge is clipped), Right, Left, Top, Bottom.

Joint data

- Type of joint. Possible values: AnkleLeft, AnkleRight, ElbowLeft, ElbowRight, FootLeft, FootRight, HandLeft, HandRight, HandTipLeft, HandTipRight, Head, HipLeft, HipRight, KneeLeft, KneeRight, Neck, ShoulderLeft, ShoulderRight, SpineBase, SpineMid, SpineShoulder, ThumbLeft, ThumbRight, WristLeft, WristRight.
- Tracking state of the current joint. Same possible values of Tracking state of the Lean body model.

- Depth space position vector, having coordinates X and Y. Named 2D space vector in Data Pre-Process and Machine Learning applications. It unit measure are the pixels (see subsection 4.4.6).
- Camera space position vector, having coordinates X, Y, and Z. Named 3D space vector in Data Pre-Process and Machine Learning applications. It unit measure are the meters (see subsection 4.4.6).
- Orientation vector, having coordinates W, X, Y, and Z. This vector is known as the quaternion representation of the joint orientation (named Vector4 in the Kinect documentation).

Data format As mentioned before, there is the possibility of storing the gesture data in different formats: 'Raw data' and 'Non-raw data'. 'Raw data' is just the data tracked by the Kinect, stored in the CSV files. 'Non-raw data' is the data tracked by the Kinect plus a header on the top of the CSV file with the names of the characteristics stored in the file. These names appear in the same order that the data tracked have so they help to have an overview of the gesture.

4.3.4 Generation of training data for evaluation

In order to test and analyze every phase of the gesture recognition, an easy gesture has been used to generate the dataset for the experimental part of the project (see section 4.6). The gesture is a clockwise circle performed by the right hand. The advantage of using that gesture is that it only involves one joint, in this case, the right wrist, is that it allows focussing more on the analysis of the data in every step of the learning path.

The training dataset is composed of 100 instances of the gesture that were performed by the author of the memory. 50 instances are 'Good' examples of the gesture, while the other 50 are bad examples. The bad examples consist of random movements with the right hand, so they have very different values than the 50 good instances.

In order to have more diversity between good instances of the gesture, it has been performed in different positions in front of the camera, with different amplitudes and durations. As far as the Kinect sensor captures data at the speed of 30 frames per second, gestures that last the same amount of seconds probably have different numbers of frames. This is the case of the dataset, where every gesture is different in terms of frames longitude.

4.4 Data pre-process application

Once the dataset is generated it is needed to apply the appropriate transformations so data can be used to train machine learning algorithms. This set of transformations is also called "pre-process" or "data pre-process" because they are transformations that process the data before the training phase. The main objective of the pre-processing is to apply the transformations proposed in *Easy gesture recognition for Kinect paper* [8], and analyze its results. However, assuming that classification algorithms only accept inputs of the same length, additional transformations must be added to achieve a dataset with gestures of the same length.

The pre-process functionality has been implemented using a Python application, which is explained in the following subsections, as well as the input, the output data, and the analysis of the transformations over the dataset.

4.4.1 Normalization and center translation

Considering that users can record the same gesture in different positions in front of the Kinect sensor, and these users may have different heights, it is needed to apply some transformation to normalize the data that is tracked. The paper *Easy gesture recognition for Kinect paper* [8] deals with this problem using two transformations:

- Moving the positions of the tracked bodies to the center of the Kinect sensor. In this project, this transformation is called 'Center translation'.
- Normalizing the heights of the tracked bodies. In this project, this transformation is called just 'Normalization'.

Obviously, both transformations only apply for joint position data, ignoring joints rotation and other features available from Kinect.

The center translation consists of the reduction of the positions of every joint using the centroid of the trajectory of the corresponding joint (see equation 4.1).

$$(x_i, y_i, z_i) = (x_i - \bar{x}, y_i - \bar{y}, z_i - \bar{z}) \tag{4.1}$$

Equation 4.1 components:

- i: frame identifier.
- (x_i, y_i, z_i): position vector of a joint in a concrete frame.
- $(\bar{x}, \bar{y}, \bar{z})$: centroid vector of a joint.

This centroid is calculated with the average of the positions of the joint over the trajectory (see equation 4.2).

$$(\bar{x}, \bar{y}, \bar{z}) = \frac{\sum_{i=1}^{n}(x_i, y_i, z_i)}{n} \tag{4.2}$$

Equation 4.2 components:

- n: number of frames.
- i: frame identifier.
- $(\bar{x}, \bar{y}, \bar{z})$: centroid vector of a joint.
- (x_i, y_i, z_i): position vector of a joint in a concrete frame.

The normalization transformation divides every position between the difference between neck and the spine base joints.

$$(x_i, y_i, z_i) = (\frac{x_i}{(x_{i,spine} - x_{i,neck})}, \frac{y_i}{(y_{i,spine} - y_{i,neck})}, \frac{z_i}{(z_{i,spine} - z_{i,neck})}) \tag{4.3}$$

Equation 4.3 components:

- i: frame identifier.
- (x_i, y_i, z_i): position vector of a joint in a concrete frame.
- $(x_{i,s}, y_{i,s}, z_{i,s})$: position vector of the spine base joint in a concrete frame.
- $(x_{i,n}, y_{i,n}, z_{i,n})$: position vector of the neck joint in a concrete frame.

These transformations can be applied in any order, but in this project they are both applied as a complete normalization transformation, being center translation executed firstly.

Equations 4.1 and 4.2 are the same that appear on *Easy gesture recognition for Kinect paper* [8]. Equation 4.3 is proposed in the paper but not detailed.

4.4.2 Segmentation

In order to have a training dataset with the same length for every gesture, it is needed to transform each gesture to a common length. The approach that has been followed is to segment each gesture in the same number of parts, where each part has the same amount of frames (it can vary in one frame depending on the length parity). These parts have been processed using two alternatives:

- Frame values being contained in the concrete part are averaged so a frame with average values is generated. The numerical values are averaged but the nominal values (like tracking state, confidence, etc) are selected using the most common value of the frames of the gesture part.
- The first frame of the part is selected and the rest of that part are discarded. This alternative conserves the original values of the frame in a sequence where frames close to the selected one have similar values due to the changes between frames are not so abrupt.

Both alternatives are considered as a good approach to obtain gestures with frames that summarize the content of the original gestures, therefore both are included in the experimentation of the project (see section 4.6).

4.4.3 Functionality and guide of use

As it was mentioned before, not only normalization transformations are important to improve the quality of the dataset, but also other transformations for this concrete problem. Transformations are also called as 'Filters'.

Here is the complete set of the filters that are available to be used with the application:

- Features filter: remove any gesture feature indicated in the filter arguments. As far as the gesture is composed in a structural way (general information, frames, and joints), removing a feature can trigger the deletion of the dependent features. This filter can target any feature of a gesture (see subsection 4.3.3) except the gesture class, the number of frames of the gesture because of these values are used in the machine learning application to implement the functionality (see section 4.5). It is internally called 'Useless filter' because it is supposed to be used to remove the gesture features that are considered useless for the training phase.

- Lasso filter: remove frames which have both hands state with the value 'Lasso'. It starts deleting from the first frame and advances reading the following ones. When the removing condition does not match again, the filter does not check the remaining frames. E.g. frames 1 to 5 contains both hands in lasso pose, but 6 to 10 (last frame) have one hand or none in lasso pose, so 1 to 5 frames are removed.

- Normalization filter: apply both center translation and normalization transformation described above.

- Include joint filter: removes the data of the joints (of every frame, see subsection 4.3.3) that are not specified in the filter arguments. The joints that are specified are not removed, being "included" in the gesture while the not specified joints are "excluded" from the gesture. It is internally called 'Joint type filter'.

- Exclude joint filter: removes the data of the joints (of every frame, see subsection 4.3.3) that are specified in the filter arguments. The joints that are specified are removed, being "excluded" from the gesture while the not specified joints are "included" in the gesture. It is internally called 'Joint type filter reverse' because is the opposite functionality that the exclude joint filter.

- Segmentation filter with original values: apply the segmentation transformation described above where a gesture is composed of frames with values of the original gesture. It is internally called 'Original segmentation filter'.

- Segmentation filter with average values: apply the segmentation transformation described above where a gesture is composed of frames with averaged values of the original gesture. It is internally called 'Average segmentation filter'.

In addition, users can decide the input dataset (indicating the location of files and the location of folders) and the location where the output dataset is generated after the transformations.

The Python application is designed to be used as a command-line program: executed in a shell with input arguments to control the execution of the transformations. Users can run this application with the transformations that they consider as appropriate, but the only specification required is the location of the input and output datasets.

A complete guide of use of this application is described in Appendix A: Data pre-process application guide of use (see chapter 9).

4.4.4 Input data

Since this application has to be executed after the Kinect Gesture Reader, the input data of the data pre-process application is specified in Gesture data (see subsection 4.3.3) and Generation of training data for evaluation (see subsection 4.3.4).

4.4.5 Output data

The output data of the pre-process of the dataset is just the same dataset but with several data transformation over it. This dataset is stored in the form of CSV files, each containing the definition of a gesture.

In order to have an evidence of where the dataset comes from, a little header is introduced in the top of each file, with the information of the parameters that were used in the pre-processing. After that information, there is a sequence of labels indicating the name of the features that the file contains, and also following the same order that the features have along the file.

Here there is the similar problem of using the name 'CSV file', that is that is not a real CSV one due to the header before the sequential data.

4.4.6 Analysis of the transformations over the training dataset

The normalization filter has been analyzed in order to know the reliability of these transformations. To achieve this, two scatter charts have been made using Microsoft Excel 2013, plotting the components of the depth space vector of the right wrist over 50 instances of the circle gesture performed by the right hand in clockwise direction. The 50 instances are labeled with the 'Good' class, being good examples of the performing of the gesture.

The depth space vector is the bi-dimensional representation of the depth image captured by the Kinect sensor. The image is mapped to a virtual screen so it unit measure are the pixels. The documentation indicates that the vector (0, 0) represents the top left corner of the screen, but looking at Figure 4.8 this vectors are located in the bottom left corner. This does not suppose any problem as far as the circle gesture looks the same upside down [64].

An alternative of analysis for the transformed data would be the camera space vector, that represents the 3D real positions captured by the Kinect [65].

Figure 4.7 shows the dataset when the normalization filter (center translation and normalization transformations) are applied.

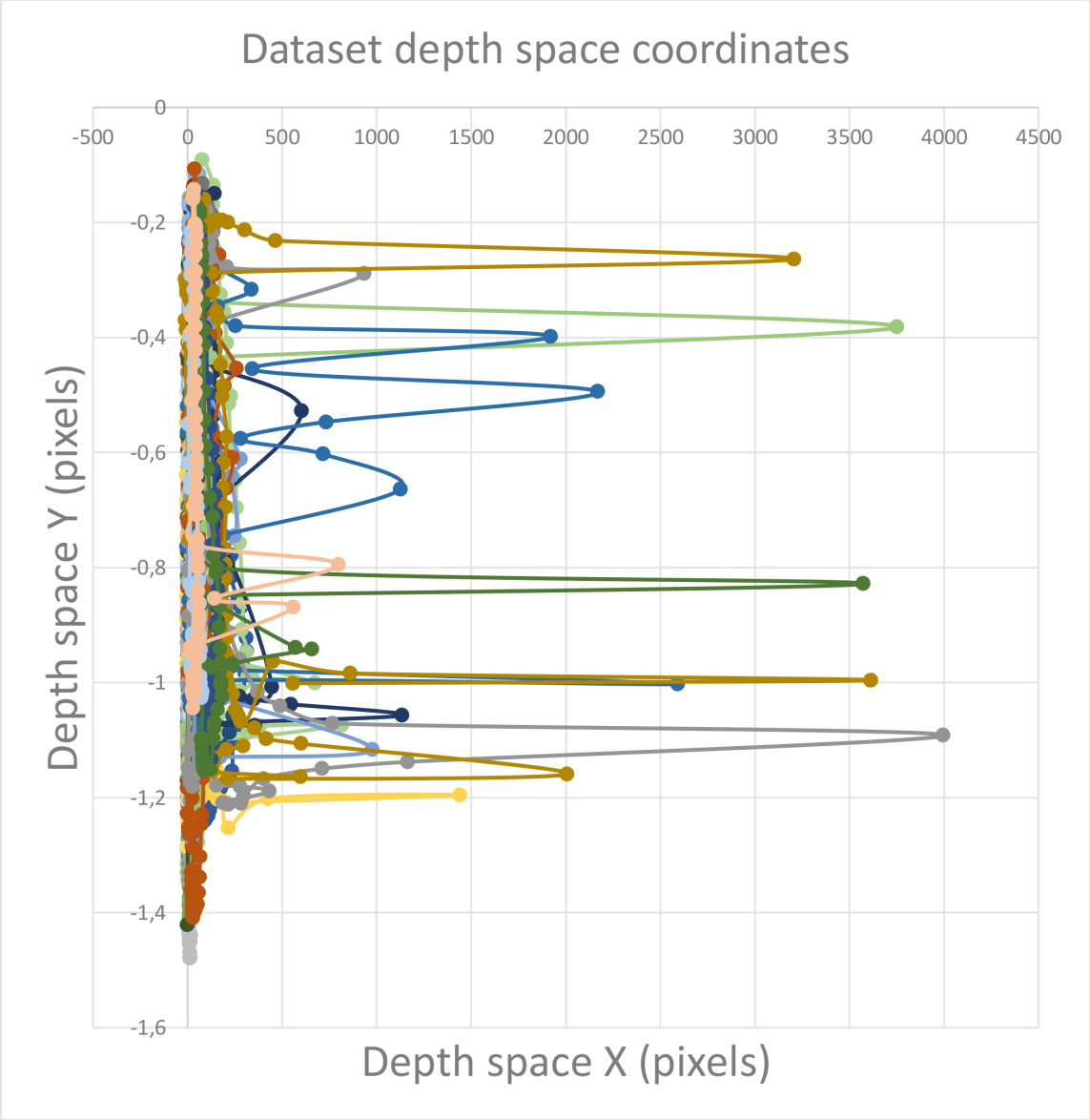

Figure 4.7: Dataset depth space coordinates with normalization.

Figure 4.8 shows the dataset when the normalization filter is not applied.

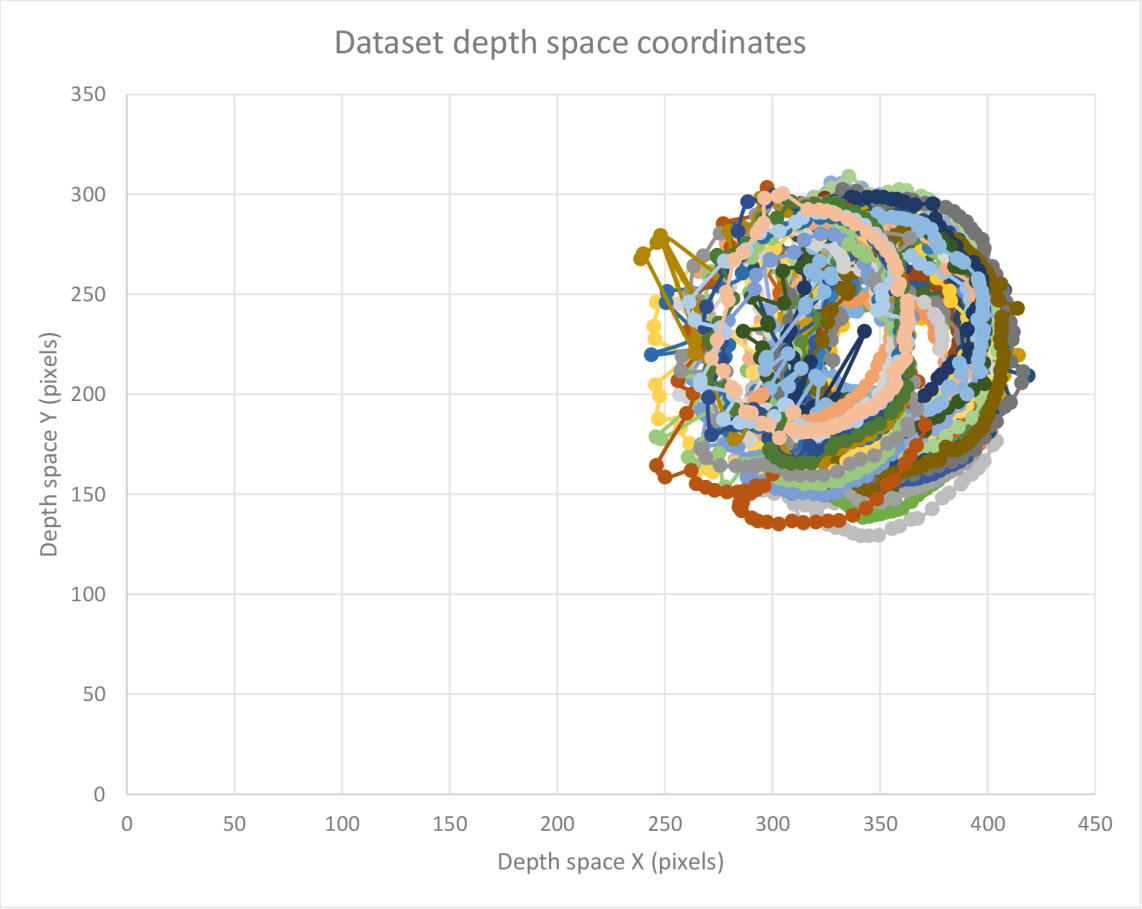

Figure 4.8: Dataset depth space coordinates without normalization.

Obviously, both graphs look very different, but the axes values of the normalized dataset increase abruptly in the case of X component and reach negatives values for the Y component. Though X component has high values in few cases, that could be considered as noise instances, this turns common to the Y component, where values are restricted between 0 and -1,5. If some of the gestures are removed from the dataset (Figure 4.9) the global figure looks more like a circle than Figure 4.7, but if it is compared with the aspect of Figure 4.8 it can be observed that normalization impact in a way that the dataset gets dirty.

However, the problem requires a normalization of the dataset so the models are independent of the heights and positions of the tracked bodies. For this reason, the experimentation contemplates both approaches, using and not using the normalization filter, to understand if this graphical representation is a good evidence of a bad normalization technique in this problem (see Section 4.6).

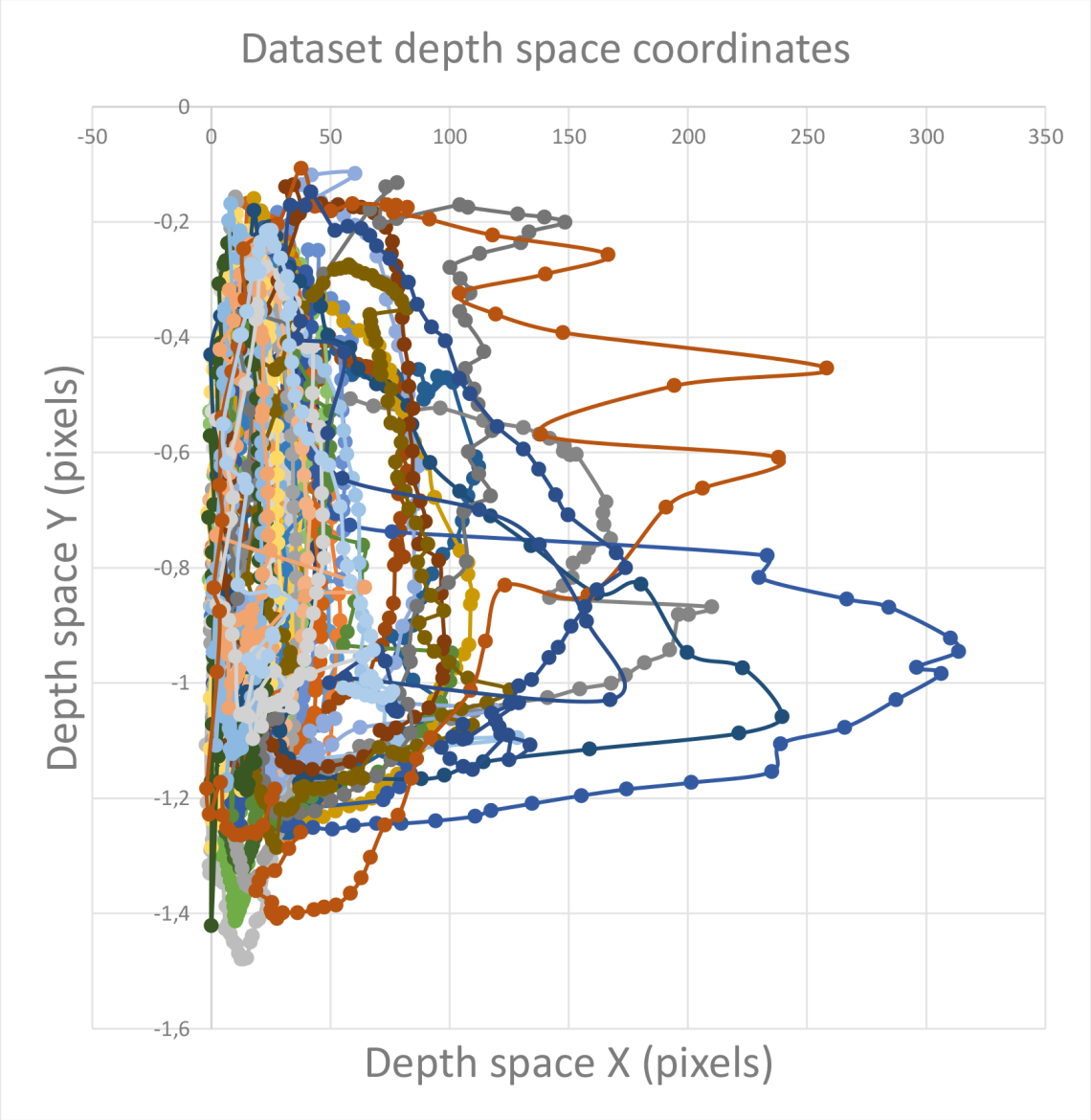

Figure 4.9: Dataset depth space coordinates with normalization but less instances.

4.5 Machine learning application

As the last step of the process, gesture recognition models are generated using machine learning algorithms. To achieve this step a Python application has been developed to apply all the algorithms so the experimentation can be done by using the three applications (see section 4.6). To avoid confusion, in this section the Python application can be also referred as 'the machine learning application'.

This section also describes the classification and clustering algorithms involved in the training process.

4.5.1 Classification

The classification is the task that recognition models have to afford in order to discriminate between Good and Bad examples of the recorded gesture (see subsection 4.3.4).

Scikit-learn library offers a large number of classification algorithms that can be used with few lines of code. In order to decide which algorithms are good to be applied to this problem, the algorithm cheat-sheet of Scikit-learn has been reviewed [66]. It recommends to use a Linear SVC if the dataset has more than 50 samples, the task is to predict a category (class), the instances of the dataset have a class (supervised learning), and the dataset has less than 100000 samples. If that classifier does not offer good results, the cheat-sheet recommends to use the K Neighbors classifier if the dataset does not have text data, or to use Naive Bayes in the case that the dataset has text data.

Nevertheless, the cheat-sheet has to be considered as a support tool to decide which of the algorithms is good to apply, so its proposals have to be analyzed previously.

Support Vector Machines Each gesture of the dataset can be represented in a space n dimensional (R^n) and a SVM classification algorithm can separate the gestures using a hyperplane $n-1$ dimensional. The better this hyperplane separates the gestures that are from a different class, the better a SVM performs the classification task. They usually work well with data with a large number of parameters, so experimentations with the complete set of features of a gesture could be afforded without so much cost [67]. However, SVM is only able to deal with numerical data, so previous data transformations are taken into account (see subsection 4.5.3).

They are considered as a good alternative because they are designed to be used for two class classifications, and Scikit-learn library offers a huge set of parameters to experiment with this algorithms. It is also possible to use different types of SVM regarding the kernel function: linear, polynomial, radial basis and sigmoid. Kernel functions are used to compare the elements of the dataset and decide the similitude between them.

K Nearest Neighbors KNN classification algorithms are also used to discriminate the class between instances that can be represented in a space n dimensional (R^n). The main difference with SMVs is that here the classification is made by checking the most common class between the k instances closest to the instances that are being classified, using the Euclidean distance to know

which are these closest instances. This algorithm is included in the group of instance-based learning because of it highly dependent on the dataset instances and does not try to generalize instead of remembering the instances that were used for training [68].

KNN is considered as a good approach for the problem because the algorithm does not have to compute a lot considering the small size of the dataset and the number of classes.

Like SVM, a transformation of the non-numerical data is required so distances between points can be compared (see subsection 4.5.3). Scikit-learn gives a large number of configuration parameters too.

4.5.2 Clustering

The clustering algorithm groups the dataset instances following some defined criterion like a distance measure between instances parameters. Each of these groups is called 'cluster', determined by a point of specific values in the space n dimensional (R^n). This representative point is called 'centroid'.

The input of the clustering algorithm is the complete set of frames of all gestures, which are grouped considering the contained characteristics. The grouping results into a set of clusters, defined by the centroid, representing a group of common values between frames. Then the algorithm assigns a cluster for each frame based on the distances and similarity between the frame and the centroid characteristics. Therefore the output is a set of gestures defined by sequences of clusters identifiers that classifiers can interpret simpler than the long sequences of features of the original gestures.

The algorithm selected to implement a clustering over the dataset is K-Means because it is one of the most used clustering algorithms across machine learning applications [69]. It is also recommended for this problem following the Scikit-learn cheat-sheet: the number of classes is known and the dataset is formed by less than 10000 instances [66].

The K-Means algorithm can be applied over the dataset but is not mandatory, so not using it is also contemplated in the experimentation (see section 4.6). Though is a ML technique, the application in this project could be considered as part of the pre-processing since it is applied to transform the training data before the classification task.

Like SVM and KNN, a transformation of the non-numerical data is required so distances between points can be compared (see subsection 4.5.3).

4.5.3 Implementation details

Numpy arrays Numpy arrays have been used in the Python application to represent the dataset structures so it is easier to be used by the classification and clustering algorithms [43].

Non-numerical data Considering that both clustering and classification algorithms need to handle numerical data, some of the gesture features defined in Gesture data (see subsection 4.3.3) need to be treated so they can be used in the machine learning application. One of the approaches that have been implemented is the One Hot Encoder transformation, which converts numerical integers into a sequence of 0s and 1s representing each value [70].

E.g. Given the following values for a certain feature: $0, 1, 2$. Value 0 is replaced by $1, 0, 0$, 1 is replaced by $0, 1, 0$ and 2 is replaced by $0, 0, 1$.

Nevertheless, that implies that non-numerical data, like the states of a hand (which uses the values: Closed, Lasso, NotTracked, Open and Unknown) has to be transformed to an integer number before applying the One Hot Encoder transformation. With the example of the states of the hand, the output vector of possible values would be 0 for Closed, 1 for Lasso, 2 for NotTracked, 3 for Open and 4 for Unknown.

The One Hot Encoder transformation is only triggered when the machine learning application detects that the input dataset has some feature that is non-numerical.

4.5.4 Functionality and guide of use

The Python application allows the configuration of a training planning over the dataset, with a series of parameters defined as input arguments (see chapter 10). The planning allows the use of the following algorithms:

- K-Means: apply clustering using K-Mean algorithm mentioned above. It is possible to select the number of clusters to be used in the algorithm. When K-Means is selected it is applied at the beginning of the application execution [71].

- SVM: apply classification using SVM algorithm mentioned above. With this algorithm it is possible to decide which type of SVM to be applied, having the option to use the C-Support Vector, the Linear Support Vector or the Nu-Support Vector [72] [73] [74]. The difference between Linear Support Vector with the rest is that the kernel function is linear and is based on a different SVM library, that supports large datasets. Nu-Support Vector offers more flexibility using the configuration of the vectors that define the hyperplane. It is also possible to define which type of kernel function to be used with the SVM, having the following alternatives: linear, polynomial, Radial Basis Function (RBF), sigmoid and precomputed.

- KNN: apply classification using KNN algorithm mentioned above. Users can define the number of neighbors of KNN, the k parameter [75].

Despite there is not a study of the consequences and implications of all the parameters of the algorithms that the machine learning application offers, some of them can be defined and then be used to study and analyze them deeply.

To extend the experimentation possibilities, users can divide the input dataset into three sub-datasets, training, test and validation, defining which percentage of the original dataset should each sub-dataset have. Training is used to train the classification algorithms so the generated models can be tested over test and validations datasets. Validation can be considered as a second test sub-dataset, since it is used to check how well the models work at the beginning of an experimentation. If the user does not assign any percentage, they will be 0% for validation, 80% for training and 20% for the test.

In addition, users can decide the input dataset (indicating the location of files and the location of folders) and the location where the results of the training and test process are stored.

The Python application is designed to be used as a command-line program: executed in a shell with input arguments to schedule the training process. Users can decide which algorithm has to be applied. It is also possible to apply clustering and not classification if the user wants to investigate further the clustering results. The only specification required is the location of the input dataset and the results.

A complete guide of use of this application is described in Appendix B: Machine learning application guide of use (see chapter 10).

4.5.5 Input data

Since this application has to be executed after the pre-processing application (see section 4.4), the input data is specified in Gesture data (see subsection 4.3.3) and Output data (see subsection 4.4.5).

4.5.6 Output data

The machine learning application generates different results depending on which algorithm is executed.

K-Means results When K-Means is executed, the application stores the following data:

- Prediction values for each of the instances, i.e. the sequence of clusters that K-Means generate assigning each frame to a cluster.
- Values of the centroids of the generated clusters.
- Distance values between each gesture and all the centroids.

Classifiers results In order to compare both classifiers, SVM, and KNN, some prediction score values are generated and stored. Considering that the dataset defined in Generation of training data for evaluation (see subsection 4.3.4) has a small size of 100 instances, it may be a wrong approach to decide which is the best classifier considering the score prediction over a test sub-dataset. Small changes in the definition of the training and test sub-datasets affect strongly to the test prediction scores. Using cross-validation and leave-one-out scores are more reliable over small datasets because they operate with a large number of mutations of the training and test datasets, and then the average of this score indicates how good the classification operates in general over all the dataset.

Cross-validation divides the dataset into k groups, and iterate k times (each iteration is usually called 'fold') so each iteration trains with the $k-1$ parts and test over the missing part. Completing all the iteration generates a vector of scores over testing. Leave-one-out is a specific of cross-validation where the k has the value of the length of the dataset.

Nevertheless, the machine learning application is designed to be used with datasets of different sizes, so the score data over test datasets are kept in the results of the classification.

Here is the complete list of the classification results that are generated and stored:

- Cross validation values: cross-validation is executed with 10 folds, so the application stores the vector with the 10 predictions values, the mean between those prediction values, and its standard deviation.
- Leave-one-out values: leave-one-out is executed and the application stores the vector with all the prediction values (depending on the size of the dataset), the mean between those prediction values, and its standard deviation.
- Prediction score over the test sub-dataset: the algorithm train with the training sub-dataset and then is tested against the test dataset. The scores of this testing are stored.
- Confusion matrix over the complete dataset: the algorithm train with the training sub-dataset and then is tested against the complete dataset. The matrix confusion of this testing is stored.

4.6 Experimentation

Once the three applications are developed, it is possible to make an experimentation planning to select which configuration of parameters fits better to create a recognizer for the clockwise circle gesture performed by the right hand.

As Generation of training data for evaluation defines (see subsection 4.3.4), the dataset is formed by 100 instances of the gesture, being 50 'Good' instances and 50 'Bad' instances. Considering that the gesture implies mainly the right hand, the dataset is pre-processed by removing all its joints except the Right Wrist (Right-Hand joint could have been selected too). In this case, the rotation does not seem to be an important feature because the gesture does not require to turn the hand in any direction. For this reason, pre-process application removes all the data except the position information, i.e. Depth space vector and Camera space vector (see subsection 4.4.6).

Regarding that it is also possible two use two types of segmentation so classification can be applied (see subsection 4.4.2), normalization must be tested (see subsection 4.4.5), there are two different positions vector and there are two classification algorithms (SVM and KNN) to be tested, the experimentation consists of the combinations of the following alternatives:

- Classification algorithm: SVM or KNN (2 values).
- K-Means execution with 8 clusters: used or not (2 values).
- Gestures attributes schemes: Depth space vector (2D), Camera space vector (3D), or its combination (3 values).
- Normalization filter: used or not (2 values).
- Segmentation type with length 8: chosen or average (2 values).

These combinations trigger the following number of configurations:

$$2 \star 2 \star 3 \star 2 \star 2 = 48$$

Both the number of clusters and the length of the segmentation have been assigned with an 8 because seems to be a reasonable segmentation of a circle gesture. This parameters can be investigated further in the case that experimentation shows bad results for all the configurations.

Figure 4.21 shows the experimentation for the SVM algorithm:

					Cross validation		**Leave one out**	
K-Means	*Attributes*	*Normalization*	*Segmentation*	*Test*	*Mean*	*Std. dev.*	*Mean*	*Std. dev.*
Yes	2D	Yes	Chosen	0,55	0,68	0,09797959	0,65	0,4769696
Yes	2D	Yes	Average	0,65	0,64	0,18	0,64	0,48
Yes	2D	No	Chosen	0,85	0,9	0,04472136	0,9	0,3
Yes	2D	No	Average	0,95	0,95	0,05	0,94	0,23748684
Yes	3D	Yes	Chosen	0,6	0,72	0,11661904	0,71	0,45376205
Yes	3D	Yes	Average	0,6	0,65	0,10246951	0,65	0,4769696
Yes	3D	No	Chosen	0,95	0,92	0,06	0,92	0,2712932
Yes	3D	No	Average	1	0,94	0,08	0,94	0,23748684
Yes	Both	Yes	Chosen	0,45	0,66	0,12	0,64	0,48
Yes	Both	Yes	Average	0,45	0,68	0,16	0,67	0,47
Yes	Both	No	Chosen	0,9	0,93	0,0781025	0,94	0,23748684
Yes	Both	No	Average	0,95	0,92	0,06	0,92	0,2712932
No	2D	Yes	Chosen	0,35	0,6	0,04472136	0,04	0,19595918
No	2D	Yes	Average	0,45	0,59	0,08306624	0,57	0,49507575
No	2D	No	Chosen	0,35	0,5	0	0	0
No	2D	No	Average	0,5	0,51	0,03	0	0
No	3D	Yes	Chosen	0,45	0,5	0	0,02	0,14
No	3D	Yes	Average	0,45	0,52	0,04	0,02	0,14
No	3D	No	Chosen	1	0,99	0,03	0,99	0,09949874
No	3D	No	Average	1	0,97	0,06403124	0,97	0,17058722
No	Both	Yes	Chosen	0,35	0,51	0,03	0	0
No	Both	Yes	Average	0,5	0,5	0	0	0
No	Both	No	Chosen	0,4	0,5	0	0	0
No	Both	No	Average	0,5	0,51	0,03	0	0

Table 4.21: SVM experimentation results.

Figure 4.22 shows the experimentation for the KNN algorithm:

					Cross validation		**Leave one out**	
K-Means	*Attributes*	*Normalization*	*Segmentation*	*Test*	*Mean*	*Std. dev.*	*Mean*	*Std. dev.*
Yes	2D	Yes	Chosen	0,75	0,66	0,14282857	0,68	0,46647615
Yes	2D	Yes	Average	0,7	0,58	0,14	0,55	0,49749372
Yes	2D	No	Chosen	0,85	0,92	0,09797959	0,95	0,21794495
Yes	2D	No	Average	0,9	0,82	0,1077033	0,84	0,36660606
Yes	3D	Yes	Chosen	0,8	0,69	0,05385165	0,72	0,44899889
Yes	3D	Yes	Average	0,65	0,66	0,10198039	0,66	0,47370877
Yes	3D	No	Chosen	0,75	0,95	0,09219544	0,95	0,21794495
Yes	3D	No	Average	0,75	0,86	0,14282857	0,85	0,35707142
Yes	Both	Yes	Chosen	0,5	0,64	0,12806248	0,61	0,48774994
Yes	Both	Yes	Average	0,45	0,68	0,13266499	0,66	0,47370877
Yes	Both	No	Chosen	0,9	0,86	0,0663325	0,85	0,35707142
Yes	Both	No	Average	0,9	0,88	0,06	0,88	0,32496154
No	2D	Yes	Chosen	0,8	0,78	0,1077033	0,77	0,42083251
No	2D	Yes	Average	0,8	0,75	0,08062258	0,74	0,43863424
No	2D	No	Chosen	0,95	0,96	0,04898979	0,96	0,19595918
No	2D	No	Average	0,85	0,96	0,0663325	0,97	0,17058722
No	3D	Yes	Chosen	0,8	0,71	0,12206556	0,73	0,44395946
No	3D	Yes	Average	0,65	0,75	0,10246951	0,76	0,42708313
No	3D	No	Chosen	1	0,98	0,04	0,99	0,09949874
No	3D	No	Average	1	0,99	0,03	0,99	0,09949874
No	Both	Yes	Chosen	0,9	0,74	0,12	0,73	0,44395946
No	Both	Yes	Average	0,9	0,76	0,11135529	0,71	0,45376205
No	Both	No	Chosen	1	0,97	0,04582576	0,96	0,19595918
No	Both	No	Average	0,9	0,97	0,04582576	0,97	0,17058722

Table 4.22: KNN experimentation results.

The test score is obtained after training the classification algorithm over the 80% of the dataset and testing the model over the other 20%. All the scores having more than 90% has been colored so they are easier to be found.

Considering the results of both experimentations, we can conclude that there a lot of configurations with great accuracy and low values of standard deviation, indicating that good results are not just a coincidence. As it was mentioned before (see subsection 4.5.6), having a small dataset to train the models can highly bias the classification results, so separating between the training and test datasets is not the better approach. This fact is illustrated by observing that the accuracy results over the test dataset are more irregular.

Cross-validation, applied with 10 folds, and leave-one-out experimentations offers more confidence in the classification results, being leave-one-out the most reliable.

The experimentation illustrates the following facts about the configuration:

- Not using K-Means shows a huge negative impact in the accuracy of SVM, while KNN has its better results with that configuration.
- The scheme of data and the segmentation choice does not seem to affect since the results between different values of that configurations are quite similar.

- The best leave-one-out results never apply the normalization transformation, so it can be considered as a bad transformation for this problem.

The best experiment of SVM shows a 99% mean accuracy of leave-one-out for the configuration:

- Do not use K-Means.
- 3D for data scheme.
- Do not use normalization
- Use the chosen segmentation

KNN shows two experiments with a 99% mean accuracy of leave-one-out, being one of them the same configuration of the best result for SVM. The other configuration only varies in the segmentation type choice.

Despite there are good results for several configurations, this configuration seems to fits better for this problem and this algorithm. Both algorithms can be considered as good alternatives to generate the recognition model but SVM seems to suffer more the impact of not using the clustering with K-Means.

Chapter 5

Conclusions and future lines

Considering the work that has been done, the three developed applications comply with the proposed functionality that is needed to generate models for gesture recognition.

The WPF application, *Kinect Gesture Reader* can be used through a UI to record customized gestures. It has been used to generate the dataset of the experimentation.

Secondly, the pre-processing application is able to apply transformation over the dataset so it can be used in the classification algorithms. The normalization and center translation transformations proposed in *Easy gesture recognition for Kinect paper* [8] are also available and has been analyzed with the generated dataset. However an overview analysis of the data (see subsection 4.4.6) and the experimentation results (see section 4.6) evidence that there is a problem with the transformation that needs additional investigation, otherwise there is only possible to make conjectures of its reason.

The machine learning application allows the generation of learning models through the classification algorithms like SVM and KNN. Also, K-Means is available to generate datasets of grouped data. The experimentation shows that using this clustering algorithm improves the accuracy in general and specially for the SVM results.

To sum up, the completion of the experimentation and the positive results obtained aims to use the complete set of developed applications to build gesture recognition models. Even so the normalization and translation formulas does not seem to fit too well in this problem, the data pre-process application is useful to apply other transformations that are essential for the training process.

5.1 Future lines

This project can be understood as a proof of concept of a future application that offers the same functionality as a whole system. For this reason, the short term future lines will be:

- Add a module for HHM in the ML tool, considering libraries like Seqlearn and Pystruct [34] [35]. This will skip the requirement of applying the segmentation filter over the dataset, as HMM is able to handle gestures with different lengths.
- Add more ML modules of classification and clustering algorithms.

- Add more parameters for every algorithm so they can be investigated further.
- Experiment with gestures that involves a higher number of joints.
- Improve the User Experience (UX) and the User Interface (UI) of the WPF application.
- Export the learning models so they can be used in any application.
- Unify the Python applications with the WPF application, so they are used as a GUI. One possible implementation would be the use of tabs to differentiate between functionalities.
- Use databases to store the datasets and their transformations over the pre-processing process.
- Investigate the normalization results of the experimentation (see section 4.6 and subsection 4.4.6) since the current experimental results are not conclusive.

All the same, in October of 2017, Microsoft announced the end of the Kinect sensor manufacturing, maintaining the support on the Xbox console and the SDK. Though Kinect technology has been applied and evolved in other Microsoft devices like Hololens or Cortana voice assistant, and Microsoft recommends the Intel's RealSense depth cameras as an alternative of Kinect device, the production discontinuation affects highly to any developer of Kinect application [54] [76]. Despite this project may not be so interesting for commercial purposes, it may support investigations over algorithms for gesture recognition.

Chapter 6

Project organization and management

This chapter describes the organization and management policies that has been followed throughout the development of the project.

6.1 Methodology

The project has been organized in three phases, regarding the three main objectives of project (see section 1.2). I have considered that avoiding a waterfall development can suppose a benefit to prevent a big effort on redefining and re-scheduling in the middle of the development process. The followed methodology that has been used is a combination of Incremental and Iterative Development [77]. To understand this methodology is important to know the meaning of each of the development approaches implied:

- Incremental: consists of developing different parts of a system in different states and assembly all of them when they are finished.
- Iterative: consists on reviewing the current work in order to improve it.

This development approach can be applied in different manners, being possible to dedicate more effort to incremental, to iterative, or both in a balanced way.

In this project I have defined three main cycles of development:

- Data capture application cycle.
- Data pre-process application cycle.
- Machine learning application cycle.

Each application is developed when the previous is finished, and the complete process can be tested when the last application is finished (Incremental development). Depending on the number of requirements, each application has different incremental cycles where each requirement can be

tested after being added (Incremental development), and then the complete application can be reviewed and improved (Iterative development).

Each single application cycle has been developed following the Personal Kanban framework, a derivative of Kanban agile framework, but focused on little teams or one single person teams [78]. It also brings more sense to this project because being one single person always generate bottlenecks between development, testing or any other phase. Personal Kanban consists of a simple board with three columns: To do, Doing, Done. To do column is also known as "Backlog", a common term in Agile frameworks referring to a stack of pending tasks. The resolution of each task increments the value of the product.

The way to fill this backlog is also important. Considering that each application was developed in order, the backlog has been filled with the tasks needed to complete the requirements (user stories) of the application that was in progress at that moment. When the backlog is empty the current application may iterate again, adding tasks to improve the application, or it may be finished so the next application can add items to the backlog.

User stories verification The tasks that are used to complete the user stories are generated with the specifications defined in the acceptance criteria of the user stories. When one of these tasks is finished it means that it has been developed and tested. Completing all the tasks of the user story state that user story as done because all its acceptance criteria has been accomplished. This logic also applies for themes and epics, e.g. if the user stories of a theme has been completed, the theme can be considered as done, because all its acceptance criteria has been completed. This has been considered the alternative of the classic testing plan after the development, complementing the methodology described here.

6.1.1 Trello

Trello has been used as the tool to follow Personal Kanban, due to its facility and natural presentation of a Kanban board [79]. It also offers a feature of adding a color label to each item, so is easier to specify categories between requirements.

6.1.2 Version control system

Github has been used as the version control system, but only for backup purposes, since there has been just one development pipeline [80].

6.2 Planning

Figures (6.1) represent the Gantt chart of the project, including the time distribution of every phase of the project. The high level design consists of the definition of the epics and themes (3.2, 3.1). It is important to indicate that Experimentation phase involves the three applications and not just the Machine Learning application.

Despite the time gap between the first month of work and the last one is a complete year, the project was developed in 210 days of part time, 4 hours per day, resulting in a total of 840 hours worked in this project. The budget of the project (see section 8.2) contemplates the real working months resulted by these number: $\frac{210 days}{30 days per month} = 7 months.$

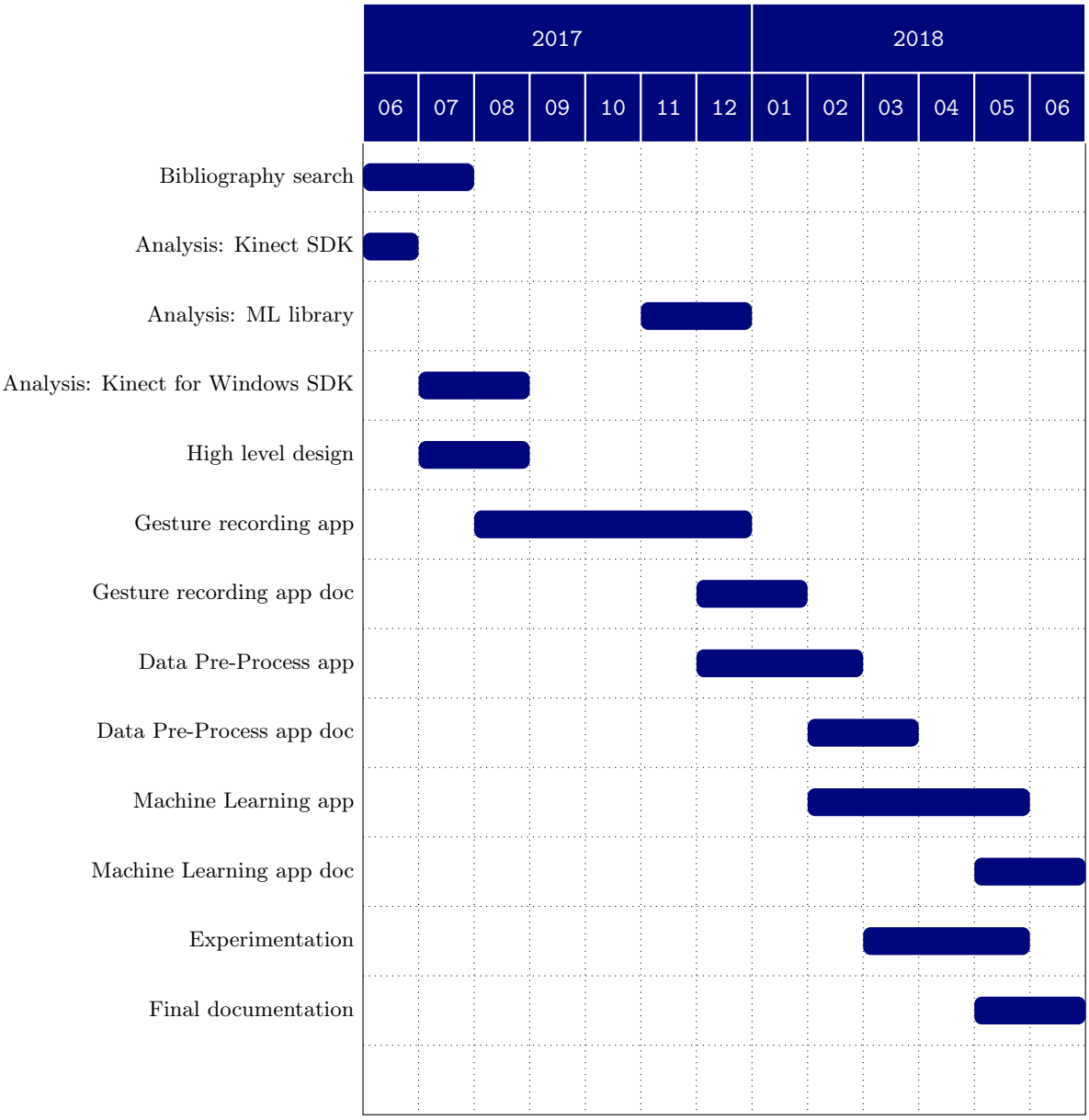

Figure 6.1: Gantt chart

Chapter 7

Regulation frame

This chapter describes the different regulations that have been contemplated and applied in the development of the project.

7.1 Legislation analysis

Privacy Since the release of Kinect sensor on Xbox 360 and then the upgraded version in Xbox One, the lack of vulnerability of user privacy has been a concern that Microsoft had to clarify. Nowadays, the Kinect and Xbox One privacy are well defined in Microsoft website. [81] [82] Users can configure which level of privacy and security they want for the Kinect experience in Xbox, being also possible to turn off the Kinect into a state which the sensor only waits for the command voice "Hey Cortana, Xbox on" to start working. Photo and video recording is available but only with the consent of the user, being also configurable which application can store data. There is another scenario where Kinect can keep user information: session data. This data is used to control applications and games, and even to improve the gaming experience. But this data, consisting of numerical values and not image or video, is stored only in the console and is removed when the gaming session finished. Kinect can also identify a user by the face points distances to sign-in into the Xbox Live system, but it is stored in the console too. Regardless, this face data cannot be used to generate a real image of the face of the user.

Though these privacy statements clarify that user privacy is not violated, they only apply when the Kinect sensor is being used with the Xbox and are not applicable using Kinect for Windows. It is necessary to check the License Terms for *Microsoft Kinect for Windows v2 Software Development Kit (SDK)*, accepted when SDK (available in [54]) is downloaded. The document of the license terms is downloaded together with the SDK itself, having the file name 'SDKEula.rf' [83]. The only privacy agreement stated in the license terms is the collection of Telemetry Data of the device using the application, but as far as the software is acquired outside the United States, Spanish law applies in the use of SDK and the developed application.

Spanish laws *LEY ORGÁNICA 15/1999, DE PROTECCIÓN DE DATOS DE CARÁCTER PERSONAL* (LOPD) and *Real Decreto 1720/2007*, that contemplates the European Regulation

General Data Protection Regulation (GDPR), has been reviewed in order to protect user personal information.

Complaining with LOPD, any user of this application must know that:

- The software stores the joint positions and angle rotation of the tracked body.
- The information is stored for the purpose of being analyzed, transformed and used to generate learning models of gesture recognition.
- The information does not contain any personal data, not even image, audio or video of the gesture performance.
- The information can be consulted, rectified and removed by the user request.

Ethical responsibilities Concerning ethical uses of the application distribution, License Terms indicates "that developers include Distributable Code in malicious, obscene, deceptive or unlawful programs;". So the same use applies to the use and distribution of the current project. The License Terms for the SDK also indicates that a Kinect application is not licensed to be used as a High-Risk Use application, due sensor and software is not designed to assure that any failure cannot happen. The project is proposed as a support tool to generate recognizers, so in case of use or distribution, this tool must not be a warranty of High-Risk Use application.

Risk of use Because of the WPF tool induces physical movements from the user, the responsible of the application using this tool must give instructions and recommendations of gesture performances, due being possible to suffer problems like overexertion or have an accident if space is not appropriate. This information is accessible in Kinect for Windows v2 Support site [84] , accessing *Product warranty and safety information* section and downloading the document *KINECT FOR WINDOWS V2 PRODUCT GUIDE* [85].

7.2 Technical standards

The WPF application development divides into the presentation part using XAML, and the back-end using C#. On May 19, 2017, the XAML Standard and .NET Standard were introduced to unify several frameworks of .NET. Given the recent state and the pending work of unification (WPF was not targeted as one of the first frameworks to adapt to the new standard), it has been considered that is better not to use this standard [86]. Nevertheless, C# back-end has been code following the coding conventions defined in Microsoft documentation [87].

The Pre-Process and the Machine Learning tools have been developed as Python scripts, so the codification has followed the Pep 8 style guide, defined in Python Developer's Guide [88].

7.3 Intellectual property

The purpose of this project is to serve as a tool to improve the development of systems with gesture recognition with Kinect v2, but it can be used as an investigation support tool. Considering that

the project consists of two types of tools, WPF and Python script, it is necessary to check the licenses applied over each of that tools so a compatible license is designed for this project.

WPF application uses the *Microsoft Kinect for Windows Software Development Kit (SDK) 2.0* so its License Terms have to be considered, besides the Microsoft .NET Library License Terms [89] [54].

Python programs use the following third party libraries:

- Scikit-learn: 3-Clause BSD license (being also called as "New BSD License" or "Modified BSD License") [90].
- NumPy: Copyright only [91].

Python code is distributed under Python Software Foundation (PSF) license.

Regarding all this licenses, the most restrictive about distribution is the Microsoft Kinect for Windows SDK 2.0 License Terms, stating that sample code can be modified, copied and distributed. This applies for the WPF application, due to it was developed through one the Kinect code samples. However, this license terms also states that the SDK software must not be published or transfered, so it has to be downloaded, installed and accessed only by the official sites available by Microsoft.

License Copyright <YEAR> <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Any person obtaining a copy of this software must respect the following License Terms of Microsoft Kinect for Windows SDK 2.0:

3. SCOPE OF LICENSE. The software is licensed, not sold. This agreement only gives you some rights to use the software. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the software only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the software that only allow you to use it in certain ways. You may not:

- *access or use, or attempt to access or use, features of the Kinect v2 Sensor that are not exposed or enabled by the software;*

- *distribute Kinect v2 Applications for use with any sensor other than the Kinect v2 Sensor or its associated drivers and runtime software;*
- *use the software or any Kinect v2 Applications in any High-Risk Use;*
- *work around any technical limitations in the software;*
- *reverse engineer, decompile, or disassemble any part of the software not provided in source code form, except and only to the extent that applicable law expressly permits, despite this limitation;*
- *publish the software for others to copy;*
- *rent, lease, or lend the software;*
- *transfer the software or this agreement to any third party; or*
- *use the software for commercial software hosting services*

The software is under licenses:

- NumPy, Copyright © 2005-2018, NumPy Developers. All rights reserved.
- Scikit-learn, 3-Clause BSD.
- Kinect for Windows SDK 2.0, Kinect for Windows SDK 2.0 License Terms
- .NET, Microsoft .NET Framework Redistributable EULA

Chapter 8

Socio-economic environment

This chapter describes the socio-economic environment of the project, considering the socio-economic impact of the developed applications and its implications on the market. Finally, the budget needed to develop the complete project is detailed.

8.1 Socio-economic impact

The set of applications developed in this project serves as a support system for developers who want to create applications for Microsoft Kinect, but it is not planned to be sold, though the Microsoft Kinect for Windows Software Development Kit (SDK) 2.0 License Terms allows its commercial use (see section 7.3). The socio-economic impact can be considered as null, but indirectly it can be used to generate recognition models that can be implemented in any application that uses Kinect v2 sensor. The state of the art of this document (see section 2.1) illustrates several application fields where this technology can be exploited.

As it is mentioned in the Futures lines of the document (see section 5.1), the Kinect production stopped recently. Once the Kinect commercial stock is empty, developers and customers will only be able to get second-hand cameras and it may be possible that they do not trust on these sensors, being forced to consider other alternatives. This fact decreases the motivation of using the set of application of the project and limits its scope to investigation purposes.

8.2 Budget

The budget for the project is divided into direct and indirect costs.

Direct cost The direct cost concern the human resources and the hardware and software needed to complete the project.

Regarding human resources, the project was developed by two persons, one of them assuming the role of a data scientist and the other the roles of a development manager, a software engineer and a test engineer. The costs of these roles are indicated in Table 8.1. As it is indicated in

planning (see section 6.2), the project was developed in 210 days (7 months) working 4 hours per day, resulting in a total of 840 hours.

Role	Salary/hour (€)	Work percentage	Worked hours	Total salary (€)
Development Manager	20,00	20	168	3.360,00
Data scientist	25,00	20	168	4.200,00
Junior Software Engineer	15,00	50	420	6.300,00
Junior Test Engineer	15,00	10	84	1.260,00
Total cost				15.120,00

Table 8.1: Human resources costs

So the total cost of the human resources of the project is **15.120,00 €**.

The hardware needed includes the Kinect v2 sensor, its adapter so the sensor can be used with a computer, and a computer compatible with the required hardware specifications detailed in *Development environment* section (see section 4.1). Table 8.2 shows the complete information of the hardware used in this project, besides other components that eased the development itself. The implicit cost of each component considers its amortization based on the duration of the project (number of months).

Component	Cost (€)	Implicit cost (€)
Microsoft Kinect Sensor v2	147,00	21,00
Microsoft Kinect Adapter	49,99	7,14
MSI CX61 2QC-1661XES computer	699,00	99,8
Nacon GM-105 Optic mouse	16,00	2,29
Logitech H110	20,00	2,86
Total cost		133,14

Table 8.2: Hardware costs

The total cost of the hardware components is **133,14 €**.

On the other hand, the Kinect for Windows SDK 2.0 and the python libraries are free of charge, and the license for Visual Studio 2015 was obtained without cost through the agreement made between the Computer Science and Engineering Department of Universidad Carlos III de Madrid and Microsoft. So the total cost of the software components is **0,00 €**.

The summarize of the direct cost is described in Table 8.3:

Concept	Cost (€)
Human resources	15.120,00
Hardware	133,14
Software	0,00
Total cost	15.253,14

Table 8.3: Direct cost summary

The total direct cost is **15.253,14 €**.

Indirect cost The indirect cost contemplated in this project is 15%, referring to any type of expense or cost that were not contemplated (risk cost).

Table 8.4 shows the total cost of the project considering the indirect cost.

Concept	Cost (€)
Direct cost	15.253,14
Risk cost	2.287,97
Total cost	17.541,11

Table 8.4: Total cost summary

So the total cost of the project would be **17.541,11 €**.

Chapter 9

Appendix A: Data pre-process application guide of use

This appendix contains the guide of use for the data pre-process application (see section 4.4). It is a Python command-line program that can be executed from a shell. Here there is an example of how it should be run:

```
python main.py -id input/ds_0 -od output/ds_0 -fuseless gesture_name total_time hand_right_confid
orientation_w -flasso -fjointtype WristRight -fsegchosen 8 -fnormal
```

This example indicates to apply the following actions:

- Use input/ds_0 as the input dataset.
- Use output/ds_0 as the output dataset.
- Remove the following features from all the gestures: the gesture name (gesture_name), the time that the gesture lasts (total_time), the right-hand confidence (hand_right_confidence) of every frame, and the W component of the orientation vector of every joint (orientation_w).
- Apply lasso filter.
- Remove all joints from every gesture except the right wrist (WristRight).
- Segment the gesture in 8 parts, using the original frame values of the gesture.
- Apply normalization filter.

The information of how to select the transformations to apply over the input dataset through the input arguments is presented in the Tables 9.2 to 9.8. In this appendix there are also included the arguments needed to indicate the input dataset and the output dataset locations (Figures 9.9, 9.10 and 9.11), though these actions are not considered as filters or transformations.

Those tables follow the Figure 9.1 format:

The meaning of each field is explained in this list:

ID	PPF-NN
Filter name	-
Flag	-
Arguments	-
Add. information	-

Table 9.1: Pre-process filter template.

- PPF-NN: is the identifier of the filter. PPF stands for Pre-Process Filter and NN identifies the filter in a sequence of all the possible filters, adopting a numerical value of two digits. This numerical value will be 01 for the first filter and is incremented with the next filter, e.g. PPF-01, PPF-02, PPF-03 (First, second and third pre-process filter).
- Filter name: name of the filter, following the description in *Functionality and guide of use* (see subsection 4.4.3).
- Flag: a string that has to be used in the Python program to apply the filter.
- Arguments: description of the arguments used in the filter.
- Add. information: gives the additional information required to understand the purpose of the filter.

ID	PPF-01
Filter name	Features filter
Flag	-fuseless
Arguments	Features identifiers separated by commas. Allowed values: gesture_name, total_frames, total_time, time_counter_max_limit, is_time_expired, frame_list, time_counter, hand_left_confidence, hand_left_state, hand_right_confidence, hand_right_state, is_restricted, joints_count, lean_x, lean_y, lean_tracking_state, tracking_id, clipped_edges, joint_list, joint_type, tracking_state, depth_space_point_x, depth_space_point_y, position_x, position_y, position_z, orientation_w, orientation_x, orientation_y, orientation_z
Add. information	None

Table 9.2: Pre-process filter PPF-01.

ID	PPF-02
Filter name	Lasso filter
Flag	-flasso
Arguments	None
Add. information	None

Table 9.3: Pre-process filter PPF-02.

ID	PPF-03
Filter name	Normalization filter
Flag	-fnormal
Arguments	None
Add. information	None

Table 9.4: Pre-process filter PPF-03.

ID	PPF-04
Filter name	Include joint filter
Flag	-fjointtype
Arguments	Joints identifiers separated by commas. Allowed values: SpineBase, SpineMid, Neck, Head, ShoulderLeft, ElbowLeft, WristLeft, HandLeft, ShoulderRight, ElbowRight, WristRight, HandRight, HipLeft, KneeLeft, AnkleLeft, FootLeft, HipRight, KneeRight, AnkleRight, FootRight, SpineShoulder, HandTipLeft, ThumbLeft, HandTipRight, ThumbRight
Add. information	Cannot be used at the same time that Exclude joint filter.

Table 9.5: Pre-process filter PPF-04.

ID	PPF-05
Filter name	Exclude joint filter
Flag	-fjointtyperev
Arguments	Joints identifiers separated by commas. Allowed values: SpineBase, SpineMid, Neck, Head, ShoulderLeft, ElbowLeft, WristLeft, HandLeft, ShoulderRight, ElbowRight, WristRight, HandRight, HipLeft, KneeLeft, AnkleLeft, FootLeft, HipRight, KneeRight, AnkleRight, FootRight, SpineShoulder, HandTipLeft, ThumbLeft, HandTipRight, ThumbRight
Add. information	Cannot be used at the same time that Include joint filter.

Table 9.6: Pre-process filter PPF-05.

ID	PPF-06
Filter name	Segmentation filter with original values
Flag	-fsegchosen
Arguments	Segmentation size.
Add. information	The segmentation size must be a positive integer number. Cannot be used at the same time that Segmentation filter with average values.

Table 9.7: Pre-process filter PPF-06.

ID	PPF-07
Filter name	Segmentation filter with average values
Flag	-fsegaverage
Arguments	Segmentation size.
Add. information	The segmentation size must be a positive integer number. Cannot be used at the same time that Segmentation filer with original values.

Table 9.8: Pre-process filter PPF-07.

ID	PPF-08
Filter name	Input dataset by file names paths
Flag	-f
Arguments	Files path, separated by spaces.
Add. information	Each file path has to be indicated regarding the location of the Python program. Each file contains the information of a single gesture. The purpose of this filter is to analyze transformation over little datasets. All the files compose the input dataset for the Python program. It is possible to use just one file.

Table 9.9: Pre-process filter PPF-08.

ID	PPF-09
Filter name	Input dataset by directory names paths
Flag	-id
Arguments	Directories paths, separated by spaces.
Add. information	Each directory path has to be indicated regarding the location of Python program. Each of these directories has files containing the information of a single gesture. All the files compose the input dataset for the Python program. It is possible to use just one directory.

Table 9.10: Pre-process filter PPF-09.

ID	PPF-10
Filter name	Output dataset by folder name path
Flag	-od
Arguments	Directory path.
Add. information	The directory path has to be indicated regarding the location of the application. The pre-processed dataset (also known as output dataset) is be saved in this directory.

Table 9.11: Pre-process filter PPF-10.

Chapter 10

Appendix B: Machine learning application guide of use

This appendix contains the guide of use for the machine learning application (see section 4.4). It is a Python command-line program that can be executed from a shell. Here there is an example of how it should be run:

```
python main.py -id input/ds_0 -od output/ds_0 -knn-num 5 -train 0.8 -test 0.2
```

This example indicates to apply the following actions:

- Use input/ds_0 as the input dataset.
- Use output/ds_0 as the output location for the results.
- Apply KNN with 5 neighbors (k parameter), using the 80% of the dataset for training, and the 20% of the dataset for testing.

The information of how to specify the parameter of the algorithms to apply over the input dataset through the input parameter is presented in the Tables 10.2 to 10.8. In this appendix there are also included the parameters needed to indicate the input dataset and the output results locations (Figures 10.9, 10.10 and 10.11), though these actions are not strictly related with the machine learning task.

Those tables follow the Figure 10.1 format:

ID	MLP-NN
Parameter name	-
Flag	-
Arguments	-
Add. information	-

Table 10.1: Machine learning parameter template.

The meaning of each field is explained in this list:

- MLP-NN: is the identifier of the parameter. MLP stands for Machine Learning Parameter and NN identifies the parameter of the application in a sequence of all the possible parameters, adopting a numerical value of two digits. This numerical value will be 01 for the first parameter and is incremented with the next parameter, e.g. MLP-01, MLP-02, MLP-03 (First, second and third parameters).
- Parameter name: name of the parameter, following the description in *Functionality and guide of use* (Subsection 4.5.4).
- Flag: a string that has to be used in the Python program to use the parameter.
- Arguments: description of the arguments used in the machine learning parameter. This may seem confusing because in some contexts argument and parameter mean the same, but here the argument is understood as the argument of a parameter of the machine learning application.
- Add. information: gives the additional information required to understand the purpose of the parameter.

ID	MLP-01
Parameter name	Number of clusters of K-Means
Flag	-cluster-num
Arguments	Number of clusters of K-Means
Add. information	The number of clusters must be a positive integer number. The clustering algorithm, in this case, K-Means is applied before any classification algorithm.

Table 10.2: Machine learning parameter `MLP-01`.

ID	MLP-02
Parameter name	Type of the SVM.
Flag	-svc-type
Arguments	Type of SVM classifier to be used as the classification algorithm. Allowed values: c, nu, linear.
Add. information	Cannot be used if a KNN parameter is also defined.

Table 10.3: Machine learning parameter `MLP-02`.

ID	MLP-03
Parameter name	Kernel of the SVM.
Flag	-svc-kernel
Arguments	Type of kernel function to be used in the SVM. Allowed values: linear, poly, rbf, sigmoid, precomputed.
Add. information	Cannot be used if a KNN parameter is also defined.

Table 10.4: Machine learning parameter `MLP-03`.

ID	MLP-04
Parameter name	Number of neighbors of KNN
Flag	-knn-num
Arguments	Number of neighbors of KNN (k parameter)
Add. information	The number of neighbors must be a positive integer number. Cannot be used if a SVM parameter is also defined.

Table 10.5: Machine learning parameter MLP-04.

ID	MLP-05
Parameter name	Training percentage
Flag	-train
Arguments	Training percentage
Add. information	The training percentage must be a positive float number, between 0 and 1. Percentages sum of training, test and validation percentages must be 1.

Table 10.6: Machine learning parameter MLP-05.

ID	MLP-06
Parameter name	Test percentage
Flag	-test
Arguments	Test percentage
Add. information	The test percentage must be a positive float number, between 0 and 1. Percentages sum of training, test and validation percentages must be 1.

Table 10.7: Machine learning parameter MLP-06.

ID	MLP-07
Parameter name	Validation percentage
Flag	-valid
Arguments	Validation percentage
Add. information	The validation percentage must be a positive float number, between 0 and 1. Percentages sum of training, test and validation percentages must be 1.

Table 10.8: Machine learning parameter MLP-07.

ID	MLP-08
Parameter name	Input dataset by file names paths
Flag	-f
Arguments	Files path, separated by spaces.
Add. information	Each file path has to be indicated regarding the location of the Python program. Each file contains the information of a single gesture. The purpose of this parameter is to analyze training over little datasets. All the files compose the input dataset for the Python program. It is possible to use just one file.

Table 10.9: Machine learning parameter MLP-08.

ID	MLP-09
Parameter name	Input dataset by directory names paths
Flag	-id
Arguments	Directories paths, separated by spaces.
Add. information	Each directory path has to be indicated regarding the location of Python program. Each of these directories has files containing the information of a single gesture. All the files compose the input dataset for the Python program. It is possible to use just one directory.

Table 10.10: Machine learning parameter `MLP-09`.

ID	MLP-10
Parameter name	Output dataset by folder name path
Flag	-od
Arguments	Directory path.
Add. information	The directory path has to be indicated regarding the location of the application. The results of the training and test process are saved in this directory.

Table 10.11: Machine learning parameter `MLP-10`.

References

[1] *Gesture recognition Definition from PC Magazine Encyclopedia*. [Online]. Available: https://www.pcmag.com/encyclopedia/term/43756/gesture-recognition (visited on 05/25/2018).

[2] *What is Gesture Recognition? - Definition from Techopedia*, en. [Online]. Available: https://www.techopedia.com/definition/618/gesture-recognition (visited on 05/25/2018).

[3] *What is gesture recognition? - Definition from WhatIs.com*, en. [Online]. Available: https://whatis.techtarget.com/definition/gesture-recognition (visited on 05/25/2018).

[4] *What is Human-Computer Interaction (HCI)?*, en. [Online]. Available: https://www.interaction-design.org/literature/topics/human-computer-interaction (visited on 05/25/2018).

[5] M. Turk, "Perceptual User Interfaces", en, in *Frontiers of Human-Centered Computing, Online Communities and Virtual Environments*, Springer, London, 2001, pp. 39–51, ISBN: 978-1-4471-1069-9 978-1-4471-0259-5. DOI: 10.1007/978-1-4471-0259-5_4. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4471-0259-5_4 (visited on 05/25/2018).

[6] R. Watson, "A Survey of Gesture Recognition Techniques", en, Trinity College Dublin, Department of Computer Science, Technical Report, Jul. 1993. [Online]. Available: http://www.tara.tcd.ie/handle/2262/12658 (visited on 05/27/2018).

[7] V. Pterneas, *Implementing Kinect gestures*, en-US, Jan. 2014. [Online]. Available: https://pterneas.com/2014/01/27/implementing-kinect-gestures/ (visited on 05/30/2018).

[8] R. Ibañez, Á. Soria, A. Teyseyre, and M. Campo, "Easy gesture recognition for Kinect", *Advances in Engineering Software*, vol. 76, pp. 171–180, Oct. 2014, Restringed access, ISSN: 0965-9978. DOI: 10.1016/j.advengsoft.2014.07.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997814001161 (visited on 05/16/2018).

[9] *Touchless Multifactor Authentication (MFA) for Healthcare*, en. [Online]. Available: https://www.intel.com/content/www/us/en/healthcare-it/solutions/documents/touchless-multifactor-authentication-healthcare-paper.html (visited on 05/27/2018).

[10] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based Hand-gesture Applications", *Commun. ACM*, vol. 54, no. 2, pp. 60–71, Feb. 2011, ISSN: 0001-0782. DOI: 10.1145/1897816.1897838. [Online]. Available: http://doi.acm.org/10.1145/1897816.1897838 (visited on 05/27/2018).

[11] G. a. S. T. N. F. Trust, *First for touchless technology in vascular surgery*, en. [Online]. Available: https://www.guysandstthomas.nhs.uk/Home.aspx (visited on 05/29/2018).

[12] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine Recognition of Human Activities: A Survey", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1473–1488, Nov. 2008, http://www.public.asu.edu/ pturaga/papers/survey_final.pdf, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2008.2005594.

[13] Z. Zhang, "Microsoft Kinect Sensor and Its Effect", en, *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, Feb. 2012, ISSN: 1070-986X. DOI: 10.1109/MMUL.2012.24. [Online]. Available: http://ieeexplore.ieee.org/document/6190806/ (visited on 05/28/2018).

[14] *Kinect in Physiotherapy - Home*. [Online]. Available: https://i.cs.hku.hk/fyp/2012/fyp12013/index.html (visited on 05/28/2018).

[15] S. Mitra and T. Acharya, "Gesture Recognition: A Survey", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, May 2007, ISSN: 1094-6977. DOI: 10.1109/TSMCC.2007.893280. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.6243&rep=rep1&type=pdf.

[16] R. Schiavullo, *SoftKinetic: Depth sensor, camera and modules for VR human body tracking for gaming and Car Infotainment Gesture Control*, en-GB, Nov. 2017. [Online]. Available: http://twittertechnews.com/virtualreality/softkinetic-depth-sensor-camera-and-modules-for-vr-human-body-tracking-for-gaming-and-car-infotainment-gesture-control/ (visited on 05/29/2018).

[17] *Sony Depthsensing Solutions > Depthsense > Depthsense Middleware*. [Online]. Available: https://www.sony-depthsensing.com/DepthSense/DepthSenseMiddleware (visited on 05/29/2018).

[18] *Kinect Hacks - Supporting the Kinect Hacking news and community*, en-US. [Online]. Available: http://www.kinecthacks.com/ (visited on 05/29/2018).

[19] patseer, *Patent Landscape Report Hand Gesture Recognition PatSeer Pro*, en-US. [Online]. Available: https://patseer.com/2017/10/patent-landscape-report-hand-gesture-recognition-patseer-pro/ (visited on 05/29/2018).

[20] *What is RGB-D Camera (Depth Sensor) | IGI Global*. [Online]. Available: https://www.igi-global.com/dictionary/human-motion-analysis-and-simulation-tools/50427 (visited on 06/05/2018).

[21] *Vitruvius | Create stunning Kinect apps in minutes*, en-US. [Online]. Available: https://vitruviuskinect.com/ (visited on 06/05/2018).

[22] *Vitruvius: A set of easy-to-use Kinect utilities that will speed-up the development of your projects*, original-date: 2013-12-03T14:57:05Z, May 2018. [Online]. Available: https://github.com/LightBuzz/Vitruvius (visited on 06/05/2018).

[23] *Vangos Pterneas | Kinect & Hololens*, en-US. [Online]. Available: https://pterneas.com/ (visited on 06/05/2018).

[24] *GesturePak v2.0 for Kinect for Windows 2.0*. [Online]. Available: http://www.franklins.net/ (visited on 06/05/2018).

[25] *Gesture Recognition Toolkit — NickGillianWiki*. [Online]. Available: http://www.nickgillian.com/wiki/pmwiki.php?n=GRT.GestureRecognitionToolkit (visited on 06/05/2018).

[26] *Nick Gillian*. [Online]. Available: http://www.nickgillian.com/ (visited on 06/05/2018).

[27] T. M. (LLC), *Visual Gesture Builder (VGB)*, en-us. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn785304(v%3dieb.10) (visited on 06/05/2018).

[28] ——, *Kinect Studio*, en-us. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn785306(v%3dieb.10) (visited on 06/05/2018).

[29] *The Discipline of Machine Learning*, en. [Online]. Available: https://www.researchgate.net/publication/268201693_The_Discipline_of_Machine_Learning (visited on 05/30/2018).

[30] *Scientific Computing Tools for Python — SciPy.org*. [Online]. Available: https://www.scipy.org/about.html (visited on 06/05/2018).

[31] *Mlpy - Machine Learning Python*. [Online]. Available: http://mlpy.sourceforge.net/ (visited on 06/01/2018).

[32] *Orange – Data Mining Fruitful & Fun*. [Online]. Available: https://orange.biolab.si/ (visited on 06/01/2018).

[33] *TensorFlow*, en. [Online]. Available: https://www.tensorflow.org/ (visited on 06/01/2018).

[34] *Seqlearn: Sequence classification library for Python — seqlearn 0.1 documentation*. [Online]. Available: http://larsmans.github.io/seqlearn/ (visited on 06/01/2018).

[35] *PyStruct - Structured Learning in Python — pystruct 0.2.4 documentation*. [Online]. Available: http://pystruct.github.io/ (visited on 06/01/2018).

[36] *Scikit-learn: Machine learning in Python — scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/index.html (visited on 06/01/2018).

[37] *Numl*. [Online]. Available: http://numl.net/ (visited on 06/01/2018).

[38] *AForge.NET :: Computer Vision, Artificial Intelligence, Robotics*. [Online]. Available: http://www.aforgenet.com/ (visited on 06/01/2018).

[39] *Accord.NET Machine Learning Framework*, en. [Online]. Available: http://accord-framework.net/ (visited on 06/01/2018).

[40] *Encog Machine Learning Framework*. [Online]. Available: https://www.heatonresearch.com/encog/index.html (visited on 06/01/2018).

[41] *Machine Learning | Microsoft Azure*, en. [Online]. Available: https://azure.microsoft.com/en-us/services/machine-learning-studio/ (visited on 06/01/2018).

[42] *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/ (visited on 06/01/2018).

[43] *NumPy — NumPy*. [Online]. Available: http://www.numpy.org/ (visited on 06/01/2018).

[44] *Python Data Analysis Library — pandas: Python Data Analysis Library*. [Online]. Available: https://pandas.pydata.org/ (visited on 06/01/2018).

[45] dotnet-bot, *Windows Presentation Foundation*, es-es. [Online]. Available: https://docs.microsoft.com/es-es/dotnet/framework/wpf/ (visited on 06/05/2018).

[46] *Frequently Asked Questions — scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/faq.html#why-did-you-remove-hmms-from-scikit-learn (visited on 06/05/2018).

[47] *WiiBrew*. [Online]. Available: http://wiibrew.org/wiki/Main_Page (visited on 06/05/2018).

[48] J. C. Lee, "Hacking the Nintendo Wii Remote", *IEEE Pervasive Computing*, vol. 7, no. 3, pp. 39–45, Jul. 2008, ISSN: 1536-1268. DOI: 10.1109/MPRV.2008.53.

[49] *(1) Gesture Recognition with a Wii Controller*, en. [Online]. Available: https://www.researchgate.net/publication/30012906_Gesture_Recognition_with_a_Wii_Controller (visited on 06/05/2018).

[50] *Wiigee - a java-based gesture recognition library for the Wii remote*. [Online]. Available: http://www.wiigee.com/ (visited on 06/05/2018).

[51] *Sony announces Move.me application for researchers and hobbyists, promises improvements to PlayStation Home*, en. [Online]. Available: https://www.engadget.com/2011/03/03/sony-announces-move-me-application-for-researchers-and-hobbyists/ (visited on 06/05/2018).

[52] [Online]. Available: https://store.playstation.com/en-us/product/UP9002-NPUO00014_00-MOVEMESERVER0000 (visited on 06/05/2018).

[53] *Sony Acquires Belgian Innovator of Range Image Sensor Technology, Softkinetic Systems S.A., in its Push Toward Next-Generation Range Image Sensors and Solutions*, en. [Online]. Available: http://www.sony.net/SonyInfo/News/Press/201510/15-083E/index.html (visited on 06/05/2018).

[54] *Kinect - Windows app development*. [Online]. Available: https://developer.microsoft.com/en-us/windows/kinect (visited on 06/05/2018).

[55] *Leap Motion*, en-US. [Online]. Available: https://www.leapmotion.com/ (visited on 06/05/2018).

[56] *Leap Motion Developers*. [Online]. Available: https://developer.leapmotion.com/documentation/index.html?proglang=current (visited on 06/05/2018).

[57] *Get Started with Our SDK*, en-US. [Online]. Available: http://developer.leapmotion.com/get-started/ (visited on 06/05/2018).

[58] *Documentation.* [Online]. Available: https://developers.usens.com/docs (visited on 06/05/2018).

[59] *NUITRACK.* [Online]. Available: https://nuitrack.com/.

[60] *Nuitrack: Overview.* [Online]. Available: http://download.3divi.com/Nuitrack/doc/Overview_page.html (visited on 06/05/2018).

[61] *(1) Master Thesis: SAVKiT: Visualization and Multimodal Manipulation of Stereographic 3-D Objects.* en. [Online]. Available: https://www.researchgate.net/publication/309349400_Master_Thesis_SAVKiT_Visualization_and_Multimodal_Manipulation_of_Stereographic_3-D_Objects (visited on 06/14/2018).

[62] *JointType Enumeration.* [Online]. Available: https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx (visited on 06/14/2018).

[63] *Body Class.* [Online]. Available: https://msdn.microsoft.com/en-us/library/microsoft.kinect.body.aspx (visited on 06/14/2018).

[64] T. M. (LLC), *DepthSpacePoint Structure*, en-us. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn758559(v%3dieb.10) (visited on 06/15/2018).

[65] ——, *CameraSpacePoint Structure*, en-us. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn758354(v%3dieb.10) (visited on 06/15/2018).

[66] *Choosing the right estimator — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html (visited on 06/17/2018).

[67] *1.4. Support Vector Machines — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/svm.html (visited on 06/17/2018).

[68] *1.6. Nearest Neighbors — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/neighbors.html (visited on 06/17/2018).

[69] *2.3. Clustering — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/clustering.html.

[70] *Sklearn.preprocessing.OneHotEncoder — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html#sklearn.preprocessing.OneHotEncoder (visited on 06/17/2018).

[71] *Sklearn.cluster.KMeans — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans (visited on 06/17/2018).

[72] *Sklearn.svm.SVC — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (visited on 06/17/2018).

[73] *Sklearn.svm.LinearSVC — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC (visited on 06/17/2018).

[74] *Sklearn.svm.NuSVC — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html#sklearn.svm.NuSVC (visited on 06/17/2018).

[75] *Sklearn.neighbors.NearestNeighbors — scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html#sklearn.neighbors.NearestNeighbors (visited on 06/17/2018).

[76] M. Wilson and M. Wilson, *Exclusive: Microsoft Has Stopped Manufacturing The Kinect*, en-US, Oct. 2017. [Online]. Available: https://www.fastcodesign.com/90147868/exclusive-microsoft-has-stopped-manufacturing-the-kinect (visited on 06/17/2018).

[77] *Using both incremental and iterative development*, en. [Online]. Available: https://www.researchgate.net/publication/296818815_Using_both_incremental_and_iterative_development (visited on 06/08/2018).

[78] A. Henry, *Productivity 101: How to Use Personal Kanban to Visualize Your Work*, en-US. [Online]. Available: https://lifehacker.com/productivity-101-how-to-use-personal-kanban-to-visuali-1687948640 (visited on 06/08/2018).

[79] *Trello.* [Online]. Available: https://trello.com (visited on 06/08/2018).

[80] *Build software better, together*, en. [Online]. Available: https://github.com (visited on 06/08/2018).

[81] *Kinect and Xbox One privacy – Microsoft privacy.* [Online]. Available: https://privacy.microsoft.com/en-us/kinect-and-xbox-one-privacy (visited on 06/07/2018).

[82] *Xbox camera and privacy.* [Online]. Available: https://privacy.microsoft.com/en-us/xbox-camera-and-privacy (visited on 06/07/2018).

[83] M. Corporation, *Microsoft Kinect for Windows Software Development Kit (SDK) 2.0 End User License Agreement*, en-US.

[84] *Kinect for Windows v2 | Kinect Support*, en. [Online]. Available: https%3A%2F%2Fsupport.xbox.com%2Fen-US%2Fxbox-on-windows%2Faccessories%2Fkinect-for-windows-v2-info (visited on 06/07/2018).

[85] ——, *KINECT FOR WINDOWS V2 PRODUCT GUIDE.*

[86] *Introducing XAML Standard and .NET Standard 2.0*, en-US, May 2017. [Online]. Available: https://blogs.windows.com/buildingapps/2017/05/19/introducing-xaml-standard-net-standard-2-0/ (visited on 06/08/2018).

[87] BillWagner, *C# Coding Conventions (C# Programming Guide)*, en-us. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions (visited on 06/08/2018).

[88] *PEP 8 – Style Guide for Python Code*, en. [Online]. Available: `https://www.python.org/dev/peps/pep-0008/` (visited on 06/08/2018).

[89] *Microsoft .NET Framework Redistributable EULA*. [Online]. Available: `https://msdn.microsoft.com/en-us/library/ms994405.aspx` (visited on 06/08/2018).

[90] *The 3-Clause BSD License — Open Source Initiative*. [Online]. Available: `https://opensource.org/licenses/BSD-3-Clause`.

[91] *NumPy license — NumPy*. [Online]. Available: `http://www.numpy.org/license.html` (visited on 06/08/2018).